SAPIENZA
UNIVERSITÀ DI ROMA

Analisi del riconoscimento delle azioni umane basato su scheletro 3D applicando tecniche di machine learning

Ingegneria dell'informazione, informatica e statistica
Informatica

Tommaso Mattei
Matricola 1884019

Tommaso Mattei

Relatore
Prof. Danilo Avola

Correlatore
Prof. Daniele Pannone

Anno Accademico 2023/2024

Analisi del riconoscimento delle azioni umane basato su scheletro 3D applicando tecniche di machine learning
Sapienza Università di Roma

Questa tesi è stata composta con LaTeX e la classe Sapthesis.

Email dell'autore: masomattei@gmail.com

Dedicato ai
miei genitori

Tutta la saggezza umana è racchiusa in queste due parole:
attendere e sperare

— Edmond Dantès

Ringraziamenti

Innanzitutto voglio ringraziare il prof. Danilo Avola e il prof. Daniele Pannone per avermi insegnato il significato di pazienza e di duro lavoro e per avermi dato l'occasione di lavorare in un ambito così interessante.

Ringrazio i miei genitori per avermi supportato per tutto questo tempo, rimanendo a me vicini fino a questo traguardo. Desidero ringraziare anche tutti i miei parenti, troppi per essere menzionati direttamente, ma sappiate che il vostro sostegno è sentito.

Un sonoro ringraziamento va a Robert, Max, Federico e Anthony; grazie per aver reso la mia esperienza universitaria migliore, e in particolar modo ringrazio Anthony per avermi aiutato e supportato durante il lavoro di tesi. Ringrazio profondamente Alessandro, Andrea, Damiano, Frasta, Francesco, Massimiliano e Mattia; grazie a voi sono arrivato fino a qui e grazie a voi continuerò ad andare avanti.

Indice

Capitolo 1

Introduzione

Il capitolo è strutturato nel seguente modo: nella sezione 1.1 viene introdotto il contesto dell'ambito e le sue possibili applicazioni; il paragrafo 1.2 riepiloga i vari approcci utilizzati nel riconoscimento delle azioni e il lavoro che si è fatto per raggiungere lo stato dell'arte; infine, nel paragrafo 1.3 vengono presentati i contributi apportati all'argomento, così come la struttura dell'intera tesi.

1.1 Ambito e applicazioni

L'ottenimento di informazioni a partire dalla vista è il modo principale in cui ci si interfaccia con la realtà. Per quanto possa essere naturale per gli esseri umani un'attività del genere, per i computer si tratta di un processo pieno di difficoltà. La branca che si occupa di questo argomento è detta Computer Vision, un campo dell'intelligenza artificiale che basa il suo studio sulle modalità con cui gli elaboratori estraggono informazioni a partire da immagini o video.

L'argomento trattato all'interno di questa tesi è in particolare il riconoscimento, da parte del computer, delle azioni compiute dagli esseri umani: si tratta della Human Action Recognition, una sottobranca della Computer Vision.

La Human Action Recognition sta diventando un ambiente di ricerca estremamente importante, che tocca numerose branche per quanto riguarda le applicazioni alla realtà. La possibilità per un computer di riconoscere da un'immagine o, più solitamente da un video, quale tipo di azione sta venendo compiuta, è importante in vari ambiti.

Ecco alcuni esempi tra le varie applicazioni:

1. **L'interazione uomo-macchina:** come viene mostrato dagli autori in [1], l'idea di interagire con dei robot intelligenti è imprescindibile dalla capacità degli elaboratori di riconoscere i movimenti degli esseri umani. Qui, come in molti altri casi, è necessaria anche un'elevata efficienza dal punto di vista della velocità di riconoscimento.

2. **L'ambito medico:** come evidenziato in [2], il riconoscimento delle azioni può essere utilizzato per il monitoraggio degli anziani, degli esercizi in generale, ma anche per la riabilitazione. La Computer Vision già si occupa di analisi

di immagini mediche e con l'HAR il collegamento tra medicina e intelligenza artificiale è sempre più rilevante.

3. **La videosorveglianza:** in questo ambito, come viene particolarmente mostrato dagli autori in [3], il riconoscimento delle azioni può portare a una reazione automatizzata ed immediata nel caso di comportamenti ritenuti nocivi. Nel paper riportato si osserva ciò relativo a potenziali atti violenti.

4. **Lo sport:** oltre ad attività come quelle previamente menzionate, anche in ambiti, con un contesto molto diverso come lo sport [4], il riconoscimento delle azioni, per quanto possa essere meno essenziale, può e potrà apportare in futuro numerosi miglioramenti in grado di aiutare le prestazioni atletiche.

Alla luce di questi fatti è necessario scegliere quale sia la tipologia migliore di dati a cui il computer dovrà associare la corretta azione performata dall'essere umano. Questa scelta può essere effetuata riflettendo sulle problematiche che spesso complicano il riconoscimento dai video:

- La mancanza di illuminazione
- la presenza di background complessi
- la variazione nei punti di vista
- la differenza di taglia tra persone
- la velocità di esecuzione

Come mostrato da alcuni studi [5], negli ultimi anni le modalità di rappresentazione dei dati sono diventate molteplici: RGB, Scheletro 2D e 3D, mappe di profondità, infrarossi, audio, radar, ecc...; esse sono a loro volta divise in modalità visive e non visive. Tra queste ultime la più comune è certamente la modalità visiva; in particolar modo gli studi si concentrano principalmente su metodi basati su RGB o Scheletro 2D/3D. [6] [7]

Tra tutte le modalità, la più popolare in assoluto finora è stata l'RGB per via della quantità di informazioni fornite sul contesto e per la facilità di ottenimento, trattandosi di comuni video a colori. Come si andrà successivamente ad analizzare, avere una grande quantità di dati è fondamentale per ottenere dei risultati accurati con il machine learning; si basti pensare a quanti video siano pubblicamente disponibili su piattaforme come YouTube [8]. Con il passare del tempo metodi basati puramente sull'RGB continuano a essere utilizzati [9], tuttavia non riescono a rispondere appieno alle generali problematiche menzionate in precedenza.

La seconda modalità in termini di utilizzo è quella basata sullo scheletro; essa un tempo non era facilmente reperibile, ma negli ultimi anni è diventata sempre più accessibile. Questo cambiamento è dovuto allo sviluppo di sensori come il Kinect [10], in grado di rilevare lo scheletro e la profondità, ma anche di algoritmi, i quali sono in grado di estrapolare informazioni sullo scheletro a partire da normali video.[11]

Per comprendere appieno la rappresentazione dello scheletro utilizzata dai metodi è necessario distinguere tra rappresentazione 2D e 3D. Come si può comprendere dalla nomenclatura lo scheletro 2D si limita alla schematizzazione, senza offrire ulteriori

informazioni sulla profondità. Lo scheletro 3D invece associa a ogni articolazione una coordinata aggiuntiva, offrendo una comprensione spaziale migliore. Negli ultimi anni la ricerca sullo scheletro 3D è predominante [12] [13] [14], essendo aumentata la facilità di ottenimento e vantando un maggior quantitativo di informazioni.

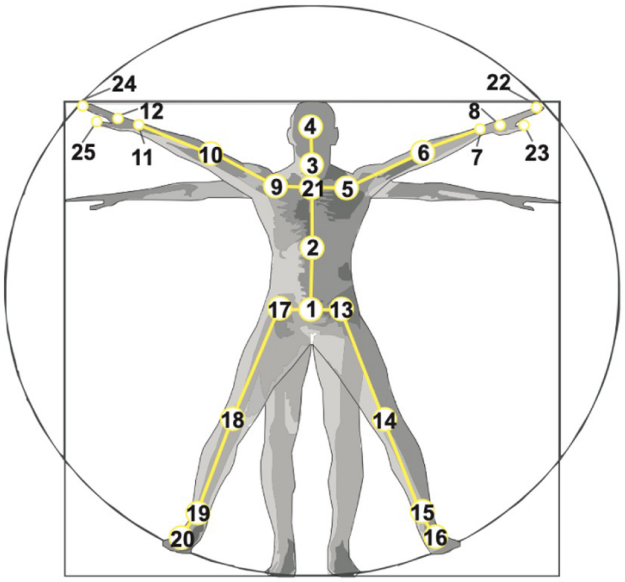

Figura 1.1. Illustrazione della rappresentazione dello scheletro umano nel dataset NTU RGB+D 120 ottenuta da [15] di Liu et al.; rappresenta schematicamente lo scheletro come articolazioni ordinate e collegate tra loro.

Comparando la rappresentazione dei dati tramite RGB con quella dello scheletro 3D è possibile comprendere il motivo della scelta di quest'ultimo per questa tesi. Sacrificando informazioni sui colori e sugli oggetti presenti nella scena, si può sopperire alla maggior parte delle problematiche previamente elencate che affliggono lo Human Action Recognition.

1.2 Stato dell'arte

Si analizzano ora i metodi di HAR utilizzati nel corso del tempo nella letteratura, relativi allo scheletro 3D, partendo dai primi approcci fino ad arrivare alle soluzioni più moderne.

1.2.1 Metodi handcrafted

Gli approcci iniziali erano principalmente basati su handcrafted features, una metodologia che prevede l'intervento diretto dell'essere umano per estrarre dalla tipologia di dati base le caratteristiche rilevanti per lo specifico problema scelto. Come menzionato in [5] ci sono due studi del tempo, che incapsulano due diverse tipologie.

Tra gli studi di questo periodo si può osservare in [17] di Wang et al. una rappresentazione dello scheletro basata sulle articolazioni (joint-based), che si basa su nuovi concetti come il Local Occupancy Pattern (LOP) e la rappresentazione con

Fourier Temporal Pyramid. Oltre a ciò introduce un modello chiamato Actionlet Ensemble Model, basato sul concetto di Actionlet: una congiunzione di features per un sottoinsieme di articolazioni rilevanti all'azione considerata.

In una direzione diversa, sempre con metodologia handcrafted, si ha in [16] di Vemulapalli et al. una rappresentazione dello scheletro basata su parti del corpo rigide, il quale usa per la classificazione una combinazione di DTW (Dynamic Time Warping), rappresentazione con Fourier Temporal Pyramid e una Macchina a vettori di supporto (SVM) lineare.

1.2.2 Metodi basati sul deep learning

Con il passare del tempo le potenzialità del machine learning – in particolar modo del deep learning – sono state comprese appieno, portandola a essere l'unica metodologia utilizzata. I risultati di questi metodi presentano i risultati migliori sui dataset più utilizzati, e si distinguono in base alla tipologia di modello sfruttata; rispettivamente: Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) e Graph-Convolutional Network (GCN).

Partendo dai metodi basati su CNN, si può osservare in [18] di Du et al. uno dei primi esempi in assoluto di un metodo che sfrutta questa architettura. Applicando feature extraction alle coordinate delle articolazioni e trasformando le informazioni ottenute in un'immagine, è possibile sfruttare questo modello di deep learning, il quale solitamente viene sfruttato per le immagini/video RGB.

Con gli anni altri modelli di questo tipo si sono evoluti, come si può vedere in [19] di Tu et al. in cui si introduce l'idea di una 3D CNN che ottiene informazioni sia spaziali che temporali, permettendo di analizzare i vari frame nel tempo.

Le 3D CNN continuano a essere rilevanti con idee innovative come in [20] di Duan et al., in cui si sfruttano delle mappe di calore associate alla CNN, riuscendo a mostrare risultati notevoli anche per l'attuale stato dell'arte.

L'altra tipologia di modello sfruttato sin dall'inizio degli studi nel riconoscimento di azioni, riguardanti il 3D Skeleton, è il Recurrent Neural Network. Il RNN è concenttualmente più vicino alla modellazione temporale necessaria nel riconoscimento delle azioni rispetto alla CNN, tuttavia esiste una correlazione tra di loro in quanto per la parte di ottenimento delle features spesso i RNN sfruttano quest'ultima tipologia.

Si può vedere in [21] di Du et al. uno dei primi esempi di RNN applicati al riconoscimento delle azioni tramite scheletro, proponendo di suddividere lo scheletro in più parti e utilizzandole come input di sottoreti neurali.

Successivamente, dati i risultati migliori e la risoluzione del vanishing/exploding gradient, una sottocategoria di RNN chiamata Long Short-Term Memory (LSTM) ha preso piede e si è affermata come migliore tipologia di modello al tempo. Si può osservare come esempio il lavoro [22] di Mahasseni et al.

Negli ultimi anni i metodi basati su RNN continuano a essere utilizzati e, come si può osservare in [23] di Friji et al., ottengono risultati molto promettenti, sfruttando approcci innovativi come quello mostrato relativo alla geometria nella fase di feature extraction.

L'ultima metodologia più comune, e quella che è emersa più recentemente, è basata su Graph-Convolutional Network (GCN). Sottocategoria del Graph Neural

Network (GNN), il GCN è divenuto il miglior metodo per il riconoscimento di azioni su 3D Skeleton per la natura stessa dello scheletro umano. Un metodo basato su grafi si presta molto facilmente a uno scheletro rappresentato da articolazioni connesse tra loro con linee, richiamando proprio la forma di un grafo.

L'introduzione del GCN avviene in [24] di Yan et al. dove una Spatial Temporal GCN riesce a modellare al meglio le interazioni con spazio e tempo; questa idea darà vita a numerose altre tipologie di GCN.

Anni dopo l'introduzione della prima GCN in questo campo numerose altre ricerche sono state effettuate come riportato in [25] di Ahmad et al. afferamandosi come il modello predominante nella letteratura. Al giorno d'oggi nuovi modelli basati su Transformers stanno emergendo [26], tuttavia il GCN rimane il modello migliore per raggiungere i migliori risultati possibili, come è possibile osservare in [27] di Myung et al..

1.3 Contributi e outline

Alla luce di tutto ciò che è stato presentato, l'obbiettivo di questa tesi è quello di partire da un determinato modello di deep learning per migliorarne le prestazioni sul riconoscimento delle azioni umane, sfruttando come tipologia di dati utlizzati lo scheletro 3D. Tra i vari modelli di deep learning sfruttati nell'ambito, il Graph-Convolutional Network mostra i risultati migliori; per questo motivo esso sarà la tipologia di architettura su cui si baserà la tesi.

La struttura del resto dei capitoli è come segue:

- **Il capitolo 2, feature extraction,** spiega cosa si intende per feature e analizza i vari metodi utilizzati in letteratura, oltre ad analizzare nello specifico le features relative allo scheletro 3D.
- **Il capitolo 3, deep learning,** è una spiegazione del machine learning a partire dall'architettura basilare, fino a raggiungere le varie tipologie di modelli previamente menzionate.
- **Il capitolo 4, architettura,** mostra nel dettaglio l'architettura del modello utilizzato in questa tesi in ogni sua componente.
- **Il capitolo 5, esperimenti e risultati,** illustra il dataset utilizzato, gli esperimenti effettuati sul modello e i risultati ottenuti.
- **Il capitolo 6, conclusioni,** parla delle conclusioni tratte dal lavoro e dei possibili sviluppi dell'argomento.

Capitolo 2

Analisi di Features

Il capitolo è strutturato nel seguente modo: nella sezione 2.1 viene introdotto il concetto di feature, spiegando nel dettaglio le metodologie associate; il paragrafo 2.2 associa alla parte teorica precedente la casistica particolare dello scheletro 3D, parlando della sua struttura, dell'ottenimento e della manipolazione delle features associate; infine, nel paragrafo 2.3 si riepiloga quanto scritto.

2.1 Features

Nel contesto della Computer Vision e del machine learning il concetto di feature è fondamentale per parlare di metodi e modelli sfruttati per la risoluzione di problemi. Per feature si intende una proprietà o carattertistica univoca che descrive il fenomeno interessato; esso solitamente possiede più features allo stesso tempo. Ad esempio per quanto riguarda un'immagine, delle features potrebbero essere alcune proprietà specifiche come i bordi presenti o i colori più utilizzati. Entrando ancor più nel dettaglio, le features si possono dividere, dal punto di vista del loro tipo, in numeriche o categoriche: le prime si presentano in formato numerico, come una data o un'altezza, mentre le seconde sono estremamente più variegate, come un nome o un colore, richiedendo infatti uno sforzo ulteriore nella loro codifica numerica. Un'ulteriore divisione che è possibile effettuare non riguarda l'essenza delle features, ma la loro relazione con l'insieme, infatti si può distinguere tra features locali e globali. Le features locali descrivono una singola regione dell'insieme in modo univoco, invece le feature globali danno una visione d'insieme del tutto, descrivendo il soggetto nella sua interezza.

Parlando di riconoscimento delle azioni degli esseri umani, le features di cui si tratterà nel resto di questa tesi sono le informazioni più importanti estratte a partire da video rappresentanti azioni. In modo più specifico si tratta delle proprietà fondamentali che si possono evincere ottenendo uno scheletro 3D, per poi successivamente transformarle in dati più succinti e accurati per il modello scelto.

Compresa la natura delle features si possono osservare e descrivere alcuni metodi relativi a esse che compongono la parte di pre-processing delle informazioni precedente al loro inserimento in un modello di machine learning. L'insieme di queste operazioni viene detto feature engineering, e ha come obbiettivo quello di portare i dati originali

in un formato adatto ai modelli di machine learning, riducendone la complessità e ottimizzandoli per ottenere predizioni migliori.

Uno dei concetti più importanti nel machine learning è quello dell'overfitting, un fenomeno in cui un modello allenato su un particolare tipo di dati non riesce a predire accuratamente dati mai visti prima. Uno dei più comuni motivi per cui ciò accade è l'inserimento in input al modello di informazioni troppo rumorose; si lavora con le features proprio per evitare questo problema.

2.1.1 Feature selection

Uno dei metodi possibili per ottenere risultati migliori con i modelli è la Feature selection o selezione delle feature. Avendo un insieme di features già ottenute nell'ambito di interesse è necessario effettuare una scelta delle stesse, avendo come obbiettivo quello di ridurre il loro numero complessivo e di scremare le features poco rappresentative. Introdurre nel proprio modello un numero troppo elevato di features non porterebbe ad altro che a una diminuzione nella velocità di allenamento ed esecuzione del modello, così come a una diminuzione nell'accuratezza, dovendo analizzare informazioni talvolta non solo inutili ma anche nocive nel complesso.

Nello scegliere quale features selezionare e quali eliminare bisogna tenere conto di alcuni principi: ridondanza delle informazioni, estraneità al problema, mancanza di informazioni, bassa varianza, ecc...

Le tre tipologie principali di algoritmi si dividono in: wrapper, filter e embedded.

1. **Wrapper:** in questa metodologia si prende l'insieme di tutte le features che si hanno a disposizione, si sceglie un sottoinsieme di features, e iterativamente si passa questo sottoinsieme al modello di machine learning, cambiando dopo ogni iterazione quale sottoinsieme viene passato. Si tratta di una metodologia computazionalmente pesante, in quanto è necessario fare una ricerca esaustiva di tutti i possibili sottoinsiemi.

   L'utilizzatore del metodo può scegliere secondo quali canoni accettare un sottoinsieme finale, ad esempio in base alla percentuale di predizione ottenuta dal modello o dal numero di features raggiunte. Un grande vantaggio del metodo è il fatto che le features finali sono perfettamente preparate per il modello particolare preso in considerazione.

2. **Filter:** questo metodo a differenza del wrapper basa la sua forza sulla velocità computazionale, sacrificando nel processo la precisione per un modello specifico. La decisione su quale sottoinsieme di features mantenere viene effettuta filtrando le features secondo un qualche tipo di legge. La legge in questione è a sé stante rispetto alla fase di machine learning e gestisce casi come la ridondanza o compara ogni features tra di loro per ottenere una lista ordinata in base all'importanza.

3. **Embedded:** infine, nei metodi Embedded si prendono elementi da ambo i metodi precedenti tentando di ottenere un compromesso tra velocità e personalizzazione. A differenza degli altri due metodi la selezione delle features è unita con l'apprendimento/costruzione del modello.

2.1.2 Feature extraction

La feature extraction o estrazione delle features è un metodo che rientra nella manipolazione delle features, risultando forse essere il più importante. Per quanto sia possibile applicare algoritmi di machine learning sulle informazioni grezze è necessario prima estrarre da esse le features più rilevanti. Mentre nel feature selection ci si limita a una scelta tra un insieme di features, qui si estrae e codifica i dati per far assumere loro una forma, spesso numerica, più consona a un modello di intelligenza artificiale. La forma in questione è detta vettore delle features (Feature Vector).

L'estrazione delle features può essere effettuato o manualmente oppure tramite algoritmi/modelli di deep learning. In un approccio manuale l'aspetto più importante da considerare è l'esperienza e conoscenza accumulata nel dominio d'interesse, portando a un'analisi accurata di quali features sono più rilevanti per il problema da affrontare.

Per quanto riguarda invece un'estrazione automatica, spesso questo compito si può affidare direttamente alla parte iniziale del modello di deep learning. Questa pratica è largamente utilizzata nello studio delle immagini e nei Convolutional Neural Network che vi si interfacciano, effettuando feature extraction internamente, avendo come input di partenza l'immagine grezza.

Oltre a quanto spiegato, spesso converge nella definizione di feature extraction anche il concetto di feature transformation. In questo lavoro si è deciso di affrontare le due parti separatamente per chiarezza esplicativa, seppure in letteratura non si tratti di una differenza così marcata.

2.1.3 Feature transformation

Il metodo di feature transformation o transformazione delle features si interessa come negli altri casi di ridurre la complessità delle features ottenute per produrre risultati migliori. In particolar modo quando si effettuano transformazioni si vuole ottenere una riduzione della dimensionalità per ottenere una comprensione maggiore delle features e una semplificazione del modello.

Transformare le features può essere fatto tramite numerosi metodi diversi; uno dei più immediati è quello basato sull'unione o combinazione di features con altre. Muovendosi oltre i problemi più banali come la ridondanza delle informazioni è evidente come nella maggior parte dei casi le features presenti nei problemi sono collegate tra di loro. Anche se con valori diversi e apparentemente non vicini tra loro, spesso sfruttando l'esperienza nel campo studiato si possono trarre conclusioni sulla correlazioni tra di esse, permettendo una fusione o manipolazione delle stesse. Per proporre un esempio: le features spaziali di un piede e un ginocchio sono numericamente diverse tra loro, tuttavia esiste una connessione logica evidente che potrebbe portare a unire le features in una sola.

Trattando la questione da un punto di vista più matematico, può essere necessario transformare le features possedute per non ottenere dei pesi sbilanciati durante il processo di allenamento del modello di machine learning. L'ordine di grandezza numerico tra i dati può essere profondamente sbilanciato, ciò provoca il rischio di ottenere dei risultati pesantemente influenzati da alcune features in particolare rispetto ad altre. Per ovviare a questo problema è necessario transformare le features

tramite una normalizzazione, un cambiamento dei dati a una scala comune che mantiene le correlazioni tra features.

2.2 Features e 3D Skeleton

Avendo introdotto il concetto di features e il tipo di analisi che si applica su di esse, si può entrare più nel dettaglio prendendo in esame la tipologia di dati scelta per questa tesi: lo scheletro 3D. Come già accennato nell'introduzione lo scheletro 3D si presenta solitamente sotto forma di punti e linee a simulare articolazioni (joints) e le ossa che le collegano; associato a ciò sono presenti delle coordinate (x, y, z) sulla loro posizione. La Figura 2.1 illustra un esempio pratico.

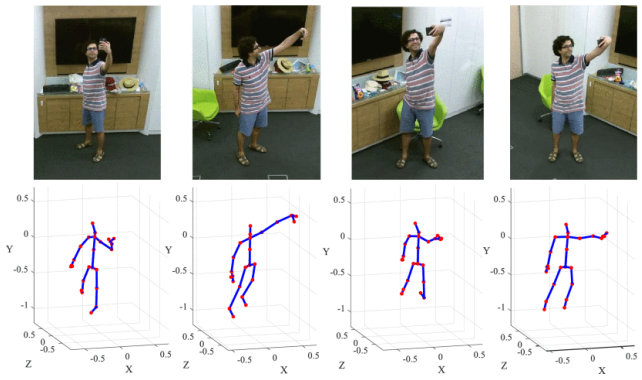

Figura 2.1. Esempio di rappresentazione di scheletro 3D in uno spazio tridimensionale, avendo associato a ogni elemento una corrispondente immagine RGB per mostrare i cambiamenti. Ottenuta da[28] di Zhang et al.

L'ottenimento stesso e la rappresentazione dello scheletro 3D fa parte del processo di feature extraction, il quale può essere diversificato in base al metodo sfruttato.

Una tra le metodologie applicate per l'ottenimento dello scheletro si basa sulla cattura di comuni video/immagini RGB, per poi applicarvi sopra un modello di deep learning in grado di estrarre autonomamente le features. La grande forza di questo approccio sta nella semplicità di ottenimento della materia prima, infatti essendo i video RGB estremamente comuni è possibile estrapolare un notevole numero di dati, i quali altrimenti sarebbero di difficile ottenimento. Alcuni esempi che rientrano in questa tipologia sono OpenPose [29] e HRNet [30].

Un'altra metodologia solitamente applicata è quella basata sulle mappe di profondità. Come menzionato in precedenza l'avvento del Kinect [31] ha permesso una facilitazione nell'ottenimento di mappe di profondità, senza dover ricorrere a ulteriori camere o tecnologie. Esistono vari approcci per l'ottenimento delle features dalle mappe di profondità, una tra le più note ricerche usa una classificazione in parti del corpo, come mostrato in [32] di Shotton et al. e come illustrato direttamente nella figura 2.2 qui sotto.

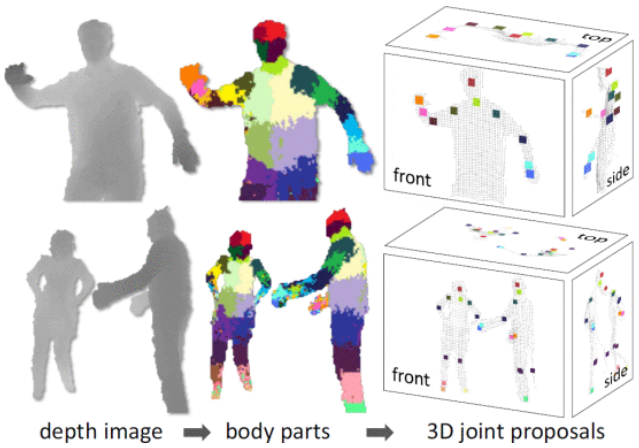

Figura 2.2. Illustrazione del passaggio da mappa di profondità a una rappresentazione basata su parti del corpo e infine a una rappresentazione scheletrica con joints. Ottenuta da[32] di Shotton et al.

In questa sezione si è trattato il concetto di features relative allo scheletro 3D e al suo ottenimento a partire dai dati grezzi. Oltre a questo primo passo, nel quale rientra la sottobranca della computer vision detta Pose Estimation, nei metodi dello stato dell'arte l'estrazione e manipolazione di features non finisce una volta estratte le informazioni sullo scheletro. L'ultimo passo prima di fornire in input le features al modello di machine learning rimane il feature transformation. La posizione spaziale, l'angolazione delle articolazioni, la velocità e la distanza tra joints; queste sono features che possono essere osservate e soprattutto manipolate nella fase finale per alleggerire ancor più il modello e per ottenere delle features più significative.

2.3 Conclusioni

Nel capitolo si è analizzato il concetto di features, spiegandone l'utilità e specificando le varie categorie di appartenenza per avere una visione d'insieme prima di entrare nel dettaglio.

Successivamente si è partiti dalla feature selection, spiegando le metodologia di scelta delle features come i metodi Wrapper, Filter e Embedded. Oltre a ciò, si è parlato di feature extraction effettuando la distinzione tra metodi manuali e automatici. Infine, si è conclusa la sezione generale discutendo del feature transformation, sfruttando le features a disposizione per ottenere informazioni più compatte e caratteristiche.

Nella seconda sezione si è scesi nello specifico, osservando le metodologie di estrazione di features relative allo scheletro 3D, distinguendo algoritmi e dati grezzi di partenza.

Capitolo 3

Deep Learning

Il capitolo è strutturato nel seguente modo: nella sezione 3.1 viene introdotto il concetto di machine/deep learning ed il suo collegamento con le reti neurali; il paragrafo 3.2 approfondisce le reti neurali partendo dal loro inizio e osservando nel dettaglio i vari step eseguiti nell'apprendimento del modello; nella sezione 3.3 si introducono dei modelli di deep learning più complessi come CNN, RNN e GCN; infine, nel paragrafo 3.4 si riepiloga ciò che è stato mostrato.

3.1 Machine learning

Il machine learning è una branca del contesto più ampio dell'intelligenza artificiale e si basa sulla costruzione e allenamento di modelli in grado di apprendere e migliorare le proprie predizioni per il riconoscimento di pattern. I modelli in questione sono in grado di compiere delle scelte autonomamente, oltre la programmazione diretta, in modo tale da poter migliorare dopo numerose iterazioni i propri pesi. L'apprendimento nel machine learning è correlato alle informazioni passate in input al modello; più precisamente è collegato alla loro quantità e qualità: esattamente ciò che ha portato al concetto di features previamente menzionato.

Le applicazioni di questa tecnologia sono numerose. Andando oltre l'argomento di questa tesi è possibile sfruttare il machine learning per l'identificazione di immagini mediche [33], per la comprensione del linguaggio (NLP) [34], motori di ricerca [35] e molto altro.

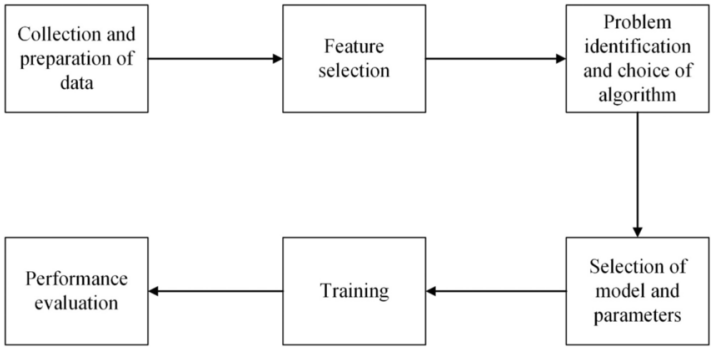

Figura 3.1. In figura una rappresentazione schematica del processo relativo al machine learning. Si inizia con i dati grezzi e l'ottenimento di features, si prosegue con l'introduzione del modello e si finisce con la performance finale dopo l'allenamento. Ottenuta da[36] di Sur et al.

Il deep learning invece è una sottocategoria del machine learning e basa la sua differenza sulla profondità della propria rete neurale e dunque sul passaggio di informazioni tra uno strato e l'altro della rete. Tutte le tipologie avanzate di machine learning (CNN, RNN, GCN, ecc...), che verranno illustrate successivamente, rientrano in questa categoria.

Delle reti neurali artificiali ora brevemente menzionate si parlerà estensivamente nella prossima sezione, in quanto l'idea e la tecnologia dietro esse è la base stessa di tutto il machine learning. Ogni modello di questa branca è una rete neurale alla base, ognuna con le sue differenze in base all'ambito e all'obbiettivo. È naturale dunque che per comprendere le forme più avanzate sia necessario comprendere queste reti basilari ed il loro funzionamento a ogni step dell'apprendimento.

3.2 Reti Neurali Artificiali

Come viene suggerito dal nome, le reti neurali artificiali sono dei modelli computazionali basati dal punto di vista concettuale sulle reti di neuroni biologiche. Dovendo affrontare la questione di una macchina in grado di apprendere e prendere decisioni autonomamente, si è deciso di prendere come riferimento il cervello umano: il modello per eccellenza per quanto riguarda l'apprendimento ed il riconoscimento di pattern.

3.2.1 Perceptron

Come mostrato nella Figura 3.2 esiste una correlazione tra un singolo neurone umano e un singolo neurone artificiale, anche detto percettrone, l'unità base della rete neurale. In un neurone biologico i dendriti si occupano dell'input di informazioni, passando per il nucleo, per poi uscire in output tramite l'assone. Allo stesso modo il percettrone riceve numerosi input i quali vengono trasformati da funzioni matematiche — di cui parleremo in dettaglio — per poi uscire in output.

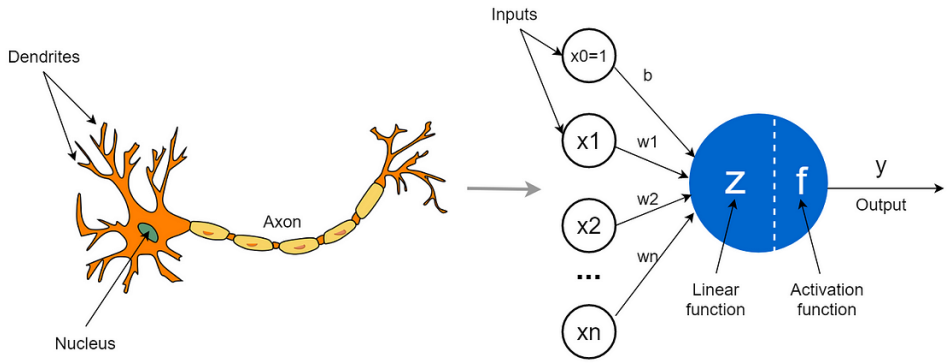

Figura 3.2. In figura un confronto tra un neurone biologico e uno artificiale. Immagine ottenuta da [37] di Pramoditha.

Un singolo percettrone è già considerabile una rete neurale, seppur si tratti del suo minimo termine. Esso rientra nella categoria di algoritmi per la risoluzione della classificazione binaria, ergo l'appartenenza o meno dell'input a una determinata classe. È possibile osservare il funzionamento su cui si basano tutte le reti neurali tramite la sua cellula più piccola; la seguente formula matematica descrive le operazioni che avvengono ai dati dall'input all'output di un singolo percettrone generico.

$$Output_j = f(w_j 0 + \sum_{i=1}^{n} w_{ji} x_i) \tag{3.1}$$

Si sta prendendo in considerazione un percettrone generico j con n input. La funzione f è detta funzione di attivazione e verrà osservata nel dettaglio in seguito; serve a produrre l'output del percettrone e comprime i risultati in un intervallo tra zero e uno. $w_j 0$ è il bias appartenente al particolare neurone; si può pensare a esso come una certa soglia da superare per l'attivazione del percettrone. Il bias viene aggiunto alla sommatoria di tutti gli input ricevuti x_i, moltiplicati per il loro peso w_{ji}, personale e diverso per ogni input in base alla provenienza. Il peso così come i bias rientrano nella categoria dei parametri da modificare nel tempo.

L'allenamento e l'apprendimento della rete neurale consistono nell'ottenimento dei giusti bias e pesi per risolvere il particolare problema posto; per fare ciò è necessario fornire una grande quantità di dati (dataset). Una parte dei dati viene utilizzata nell'allenamento della rete neurale, avvicinandosi sempre di più a generalizzare le predizioni oltre le informazioni ottenute. Un'altra parte dei dati viene sfruttata per provare il modello e osservare la percentuale di successo nel compito assegnato.

3.2.2 Multilayer Perceptron

Un Multilayer Perceptron (MLP) [38] è il naturale sviluppo dell'idea di percettrone. Così come la mente umana opera sfruttando numerosissimi neuroni, anche le reti neurali si sono sviluppate creando una struttura con numerosi percettroni collegati tra loro. A differenza della classificazione binaria del singolo percettrone, un multilayer

perceptron è in grado di classificare anche ciò che non rientra nella prima categoria. Nella Figura 3.3 si può osservare la struttura del MLP.

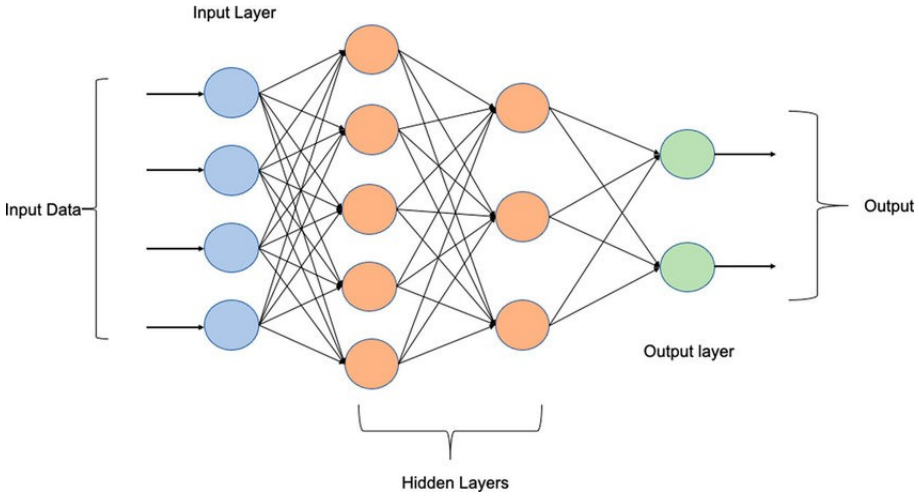

Figura 3.3. Illustrazione di un generico Multilayer Perceptron con due strati nascosti.

Il Multilayer Perceptron per definizione possiede almeno 3 layers o strati: insiemi di percettroni allineati e paralleli nella loro esecuzioni. Nel primo strato si hanno le informazioni in input, infatti è anche detto input layer. Nel mezzo della rete neurale sono presenti gli hidden layers o strati nascosti, dove l'apprendimento e la computazione avviene effettivamente; non esiste un numero massimo di questi strati ed infatti con un numero elevato di essi ci si sposta da un'architettura basilare a una di deep learning. Infine è presente l'output layer ove avviene la predizione finale.

Il Multilayer Perceptron rientra nella categoria di Feedforward Neural Network (FNN), un network dove le informazioni passano da uno strato all'altro in una sola direzione, andando sempre avanti e non tornando mai indietro (come si rivela essere in architetture più complesse come RNN).

Avendo osservato una visione d'insieme di una rete neurale, si può ora entrare nel dettaglio su ciò che è stato previamente menzionato in merito al funzionamento matematico del modello e della formula 3.1.

3.2.3 Funzione d'attivazione

La funzione d'attivazione di un percettrone prende la somma pesata degli input e dà un output ristretto in un certo raggio. Il motivo dietro il nome è che la funzione decreta l'attivazione o meno del neurone. La scelta di questa funzione varia in base al problema e può influire pesantemente sul modello risultante.

Storicamente parlando, la funzione d'attivazione più usata all'inizio è visibile nella Figura 3.3, dove viene mostrata la funzione sigmoidea, con la relativa formula 3.2.

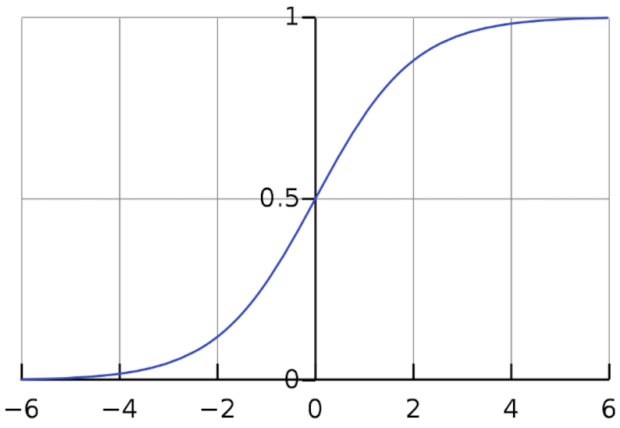

Figura 3.4. Rappresentazione della funzione sigmoidea; essa tende a zero per i valori molto inferiori a 0 mentre tende a 1 per valori molto superiori a 1.

$$sigmoid(x) = \frac{1}{1 - e^{-x}} \tag{3.2}$$

Numerose altre funzioni sono state provate ed utilizzate nel tempo, tuttavia al giorno d'oggi la funzione più comune ed efficace è la funzione ReLU, visibile nella Figura 3.4, con la relativa formula 3.3.

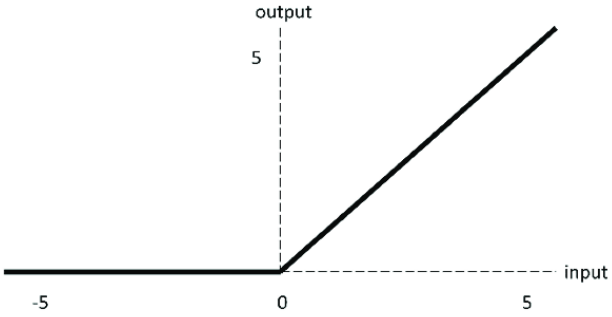

Figura 3.5. Rappresentazione della funzione Rectified Linear Unit (ReLU); essa prende il massimo tra 0 e l'input.

$$ReLU(x) = max(0, x) \tag{3.3}$$

I motivi per questa scelta sono dovuti all'efficienza e velocità di apprendimento nelle reti neurali profonde, risolvendo il problema del vanishing/exploding gradient di cui si discuterà in una delle sezioni successive. Oltre a ciò la ReLU si avvicina di più a un neurone biologico in quanto si divide più nettamente tra attivo e non attivo, portando a zero un buon numero di casi. Quando si tratta di reti neurali profonde la funzione sigmoidea non riesce a svolgere appieno il proprio compito, necessitando quindi di una sostituzione più accurata.

3.2.4 Paradigmi di apprendimento

La parte cruciale nelle reti neurali è quella relativa all'apprendimento. Per quanto il modello alla base di tutto sia sempre il multilayer perceptron, esistono vari paradigmi di apprendimento, vari approcci che mirano a modificare pesi e bias in modo diverso. Si può dividere il tutto in tre differenti metodologie:

- *Apprendimento supervisionato*: Si ricorre all'apprendimento supervisionato nel momento in cui è disponibile una grande quantità di dati, correttamente preparati per il dominio di interesse. Questo insieme di informazioni è comunemente definito dataset; esso presenta assieme ai dati anche delle etichette create dall'uomo che potranno guidare il modello nella fase di apprendimento. Grazie a queste etichette si possiede un particolare input già consci dell'output che si vuole ottenere; sarà grazie a ciò che il modello potrà assegnare ai dati conosciuti la giusta classe e potrà compiere predizioni su dati mai visti prima. La fase di testing relativa a nuovi dati è fondamentale per comprendere il livello di astrazione e la generalizzazione del modello, potendo in base alla necessità modificare gli iperparametri per ottenere diversi risultati.

- *Apprendimento non supervisionato*: In una direzione opposta rispetto all'apprendimento supervisionato, quello non supervisionato viene utilizzato nel caso una grande quantità di dati con etichette manchi, ergo non è presente alcun dataset per il compito specifico. Per sopperire a questo problema questo paradigma basa l'apprendimento della rete neurale sull'osservazione di dati senza etichetta, lasciando al modello il compito di comprendere pattern, somiglianze e una struttura nei dati grezzi. La fase di creazione del dataset dunque è insita all'interno del modello stesso, volendo infine raggruppare gli input secondo una metrica comune.

- *Apprendimenti per rinforzo*: Questa tipologia di apprendimento si distacca dalle precedenti, andando anche a toccare branche diverse dal machine learning. La tipologia di input in questo caso è sempre senza etichette, oltre a ciò si ha un interprete ed un agente. L'obbiettivo dell'apprendimento è quello di permettere all'agente di prendere azioni le quali verranno premiate e rinforzate dall'interprete, indirizzandolo verso la via più ottimale. Queste azioni avvengono all'interno di un ambiente, solitamente un Markov Decision Process (MDP).

Avendo introdotto il concetto di apprendimento, nelle prossime sezioni si osserva in ogni suo passo ciò che avviene all'interno di una rete neurale e come avviene il meccanismo secondo cui si modificano bias e pesi.

3.2.5 Funzione Costo

Rientrando nel contesto del multilayer perceptron, si è osservata la struttura della rete neurale ed il modo in cui le informazioni viaggiano fino all'output. Una volta raggiunto questo punto la fase di allenamento ha inizio; l'intero procedimento comincia con la funzione di costo.

La rete neurale inizializza i propri valori di bias e pesi in modo completamente casuale nel suo primo ciclo di allenamento, producendo un risultato di classificazione impreciso e molto lontano dal risultato ideale. Per quantificare questa imprecisione si necessita di una metrica matematica ben precisa, un numero che indichi quanto i pesi e i bias ottimali siano lontani. Il valore in questione è il risultato della funzione costo.

Qui sotto è presentata la formula matematica della funzione costo, ove h(x) è il risultato ottenuto nella singola iterazione, mentre y(x) è il risultato corretto che si vuole ottenere:

$$C = \sum_{i=1}^{m} (h(x_i) - y(x_i))^2 \tag{3.4}$$

Nell'ultimo strato della rete neurale, lo strato dell'output, bisogna considerare il risultato ottenuto meno il risultato ideale al quadrato. Questo tipo di calcolo però vale per ogni singolo neurone presente in quello strato, dunque a sua volta è necessario sommare i risultati per ogni singolo percettrone, risultando nella funzione costo.

La funzione costo si comporta nel seguente modo: più il numero ottenuto è elevato e più si è lontani dall'ottenere il risultato finale, più è vicino a zero e più si è vicini a un risultato soddisfacente.

3.2.6 Gradiente Discendente

Per comprendere in base a quali input la funzione costo diventa minima si può osservare da un punto di vista geometrico sul piano cartesiano una curva. Se si riesce a trovare il punto in cui in base all'input la funzione produce l'output minore allora si è in grado di capire come modificare i pesi e i bias.

Partendo da un punto qualsiasi all'interno della curva il modo migliore per riuscire a trovare questo minimo è osservare la "pendenza" della funzione: il minimo si troverà nella direzione opposta rispetto a dove essa sale. Questa direzione ascendente è data dal gradiente della funzione, una tipologia di vettore matematico.

Compreso il concetto di gradiente viene definito un algoritmo di ottimizzazione chiamato gradiente discendente, il quale calcola l'inverso del gradiente per capire la direzione da intraprendere per raggiungere un minimo locale della curva. Trattandosi di una rete neurale la funzione costo apparirà come uno spazio multidimensionale.

Il gradiente discendente si basa sul comprendere a ogni iterazione la direzione da intraprendere; una volta scelta si compie un passo o step in quella direzione per poi ripetere il processo, fino allo step, che porta l'algoritmo il più vicino possibile al minimo. Compiere lo step per quanto riguarda una rete neurale, implica aggiustare i bias e i pesi di una certa misura, infatti comprendere la discesa della curva dà un'intuizione su quali particolari pesi e bias abbassano il costo più rapidamente.

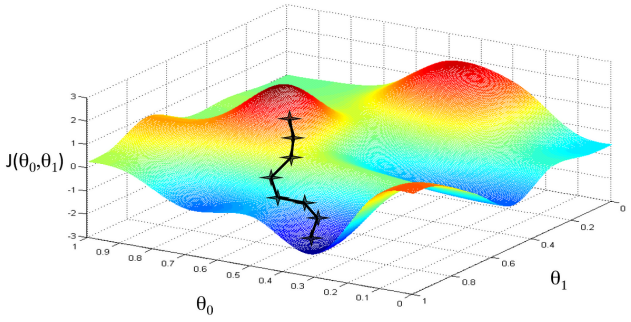

Figura 3.6. Rappresentazione del gradiente discendente per ogni passo, in uno spazio dimensionale più schematico e semplificato rispetto alla realtà.

Nello sfruttare il gradiente discendente è necessario decidere preventivamente quanto il passo compiuto dall'algoritmo a ogni iterazione sia grande. Se il passo scelto è troppo piccolo l'algoritmo sarà certamente molto preciso nel suo risultato finale, ma temporalmente parlando sarebbe inefficiente. Se il passo scelto è troppo grande l'algoritmo non riuscirà mai a raggiungere precisamente il minimo locale ottenendo un risultato mediocre.

La scelta dello step ricade sul creatore del modello, infatti il passo rientra in una categoria di parametri da inizializzare alla creazione della rete neurale, detti iperparametri. In particolar modo ci si riferisce a ciò come il learning rate o frequenza di apprendimento. Nella Figura 3.7 viene mostrato in modo più esplicativo.

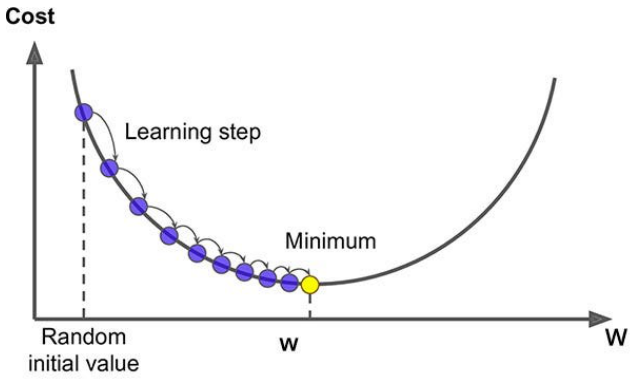

Figura 3.7. Rappresentazione del learning rate applicato a una curva con il gradiente discendente.

3.2.7 Retropropagazione dell'errore

La parte più importante e il cuore dell'apprendimento di una rete neurale è la retropropagazione dell'errore, un algoritmo in grado di sfruttare delle derivate parziali per determinare le componenti nel gradiente.

Partendo da un approccio più intuitivo, la retropropagazione rappresenta un singolo step o passo dell'algoritmo del gradiente discendente. Essa è una singola iterazione di tutti i dati d'allenamento attraverso la rete neurale, calcolando per ogni singolo esempio la media del cambiamento che esso vuole apportare a bias e pesi,

avendo come fine l'ottenimento di un particolare numero d'attivazione dei neuroni output, ergo il raggiungimento del minimo locale previamente menzionato.

Il cambiamento di bias e pesi è il gradiente della funzione costo ed è ciò che vogliamo calcolare nella retropropagazione. Il motivo per cui l'algoritmo prende questo nome è poiché dopo la fase feed-forward avviene il calcolo dei cambiamenti necessari nell'ultimo strato, e poiché essi dipendono dallo strato immediatamente precedente si invia all'indietro l'insieme di cambiamenti i quali ricorsivamente raggiungeranno il primo strato.

Compreso ciò, si può quantificare il tutto da un punto di vista più rigoroso e matematico; si comincerà a partire da formule applicate a una rete neurale composta da strati a un singolo neurone per poi generalizzare a multilayer perceptron più complessi.

Innanzitutto si ricordi la formula 3.1, la quale generalizza il calcolo dell'attivazione di un singolo neurone in una rete neurale e si definisca come $z^{(L)}$ l'attivazione di un singolo neurone allo strato L senza la funzione di attivazione (per motivi di notazione):

$$z^{(L)} = w^{(L)}a^{(L-1)} + b^{(L)} \tag{3.5}$$

Dove $w^{(L)}$ è il peso del collegamento allo strato L; $a^{(L-1)}$ è l'attivazione del neurone allo strato precedente e $b^{(L)}$ è il bias del neurone allo strato L.

Seguendo la formula, ci riferiamo all'attivazione di un neurone con la funzione di attivazione, sigmoidea per semplicità, come $a^{(L)}$:

$$a^{(L)} = \sigma(z^{(L)}) \tag{3.6}$$

Infine, ci si riferisce alla generica formula 3.4 per il costo e la si rivede applicata a questo esempio più semplice:

$$C_0 = (a^{(L)} - y)^2) \tag{3.7}$$

Dove, y è l'output ideale della rete neurale per quel neurone.

Presentate le varie annotazioni, l'obbiettivo è calcolare quanto la funzione costo cambia in base al peso, ergo una delle componenti del vettore gradiente finale. Ciò è rappresentato da $\frac{\partial C_0}{\partial w^{(L)}}$ ed è la quantificazione di quanto il peso influisca sul costo finale. Esso si calcola tramite la regola della catena, la quale mostra come il risultato sia inevitabilmente collegato allo strato precedente, ergo come sia possibile propagare all'indietro queste informazioni per gestire pesi e bias di tutta la rete neurale. Qui è presentata la formula:

$$\frac{\partial C_0}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C_0}{\partial a^{(L)}} \tag{3.8}$$

La procedura per l'equivalente dal punto di vista del bias è piuttosto simile, dovendo solo scambiare le menzioni del peso con quelle del bias:

$$\frac{\partial C_0}{\partial b^{(L)}} = \frac{\partial z^{(L)}}{\partial b^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C_0}{\partial a^{(L)}} \tag{3.9}$$

Nel contesto globale, poi, bisognerà fare la media tra tutti i risultati per ogni singolo input appartenente ai dati d'allenamento.

Ora si può osservare il caso più generale in cui la rete neurale presenta più neuroni per ogni strato. Innanzitutto riscriviamo la formula per il costo con le nuove notazioni:

$$C_0 = \sum_{j=0}^{n_L-1} (a_j^{(L)} - y_j)^2 \tag{3.10}$$

Dove, j rappresenta la posizione all'interno dello strato. Poi osserviamo la formula 3.5 in modo generico:

$$z_j^{(L)} = b_j^{(L)} + ... + w_{jk}^{(L)} a_k^{(L-1)} + ... \tag{3.11}$$

Dove, j è l'indice del neurone all'interno di un singolo strato e k è lo strato immediatamente prima di j. La notazione w_{jk} rappresenta il peso della connessione che collega il neurone nella posizione k a quello nella posizione in j, sui due strati adiacenti. Nel complesso l'intera formula rappresenta l'attivazione di un generico neurone senza funzione di attivazione.

Applicando la funzione di attivazione sigmoidea scriveremo la notazione generica $a_j^{(L)} = \sigma(z_j^{(L)})$ della formula 3.6.

Ora rivediamo genericamente la formula più importante relativa ai pesi, ottenuta tramite la regola della catena:

$$\frac{\partial C_0}{\partial w_{jk}^{(L)}} = \frac{\partial z_j^{(L)}}{\partial w_{jk}^{(L)}} \frac{\partial a_j^{(L)}}{\partial z_j^{(L)}} \frac{\partial C_0}{\partial a_j^{(L)}} \tag{3.12}$$

Esattamente come in precedenza basta sostituire al peso il bias per avere la relativa formula.

Oltre a ciò, la vera differenza nella casistica più generale è quella relativa all'attivazione dello strato L-1, nel calcolare il gradiente nello strato nascosto si ha la seguente formula:

$$\frac{\partial C_0}{\partial a_k^{(L-1)}} = \sum_{j=0}^{n_L-1} \frac{\partial z_j^{(L)}}{\partial a_k^{(L-1)}} \frac{\partial a_j^{(L)}}{\partial z_j^{(L)}} \frac{\partial C_0}{\partial a_j^{(L)}} \tag{3.13}$$

Questa modifica è dovuta al fatto che un neurone allo strato L-1 può modificare l'attivazione e funzione costo di più neuroni allo strato successivo L, rendendo necessaria una sommatoria per tutto lo strato L. Una volta ottenute queste informazioni la retropropagazione continua, potendo ricorsivamente eseguire i calcoli fino a passare per ogni singolo strato della rete neurale.

All'inizio di questa sezione si è menzionato come un singolo passo del gradiente discendente fosse applicato all'interezza dei dati d'allenamento quando nella realtà raramente è così. Per ottimizzare notevolmente la velocità di elaborazione mantenendo comunque una rappresentatività dell'intero insieme di dati, si ricorre a una variazione del gradiente discendente: il gradiente discendente stocastico.

Secondo questa metodologia si ottengono dei sottoinsiemi più ristretti di dati e si esegue una singola iterazione sopra quella singola parte, riducendo ampiamente il tempo totale necessario a raggiungere il minimo locale. Ovviamente non osservando

i dati nella loro totalità i passi compiuti dall'algoritmo saranno più imprecisi, ma alla fine convergeranno più velocemente verso il risultato.

3.3 Modelli di deep learning

Analizzata la struttura base di una rete neurale si passa a osservare architetture più complesse con numerosi strati nascosti, le stesse utilizzate nello stato dell'arte per il riconoscimento delle azioni umane con lo scheletro 3D: Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) e Graph-Convolutional Network (GCN).

3.3.1 Convolutional Neural Network

Una rete neurale convoluzionale è una tipologia di rete neurale sfruttata solitamente per il riconoscimento delle immagini, infatti la concezione del modello, in modo simile al multilayer perceptron, prende spunto dal modo in cui i neuroni gestiscono le informazioni in arrivo dall'occhio negli animali. Per quanto la CNN si presti significativamente alla classificazione di immagini, è possibile sfruttarla anche nell'ambito del riconoscimento delle azioni che non sfrutta immagini RGB. Pressoché qualsiasi tipo di modalità in cui i dati appaiono o vengono organizzati può essere con le giuste modifiche transformata in un'immagine; infatti a partire dallo scheletro 3D numerosi approcci nella letteratura sfruttano questa idea.

Essendo una rete neurale profonda la CNN presenta numerosi strati con funzioni profondamente diverse tra loro, partendo da una fase di feature extraction interna al modello fino a raggiungere la classificazione vera e propria. Ora si analizzerà l'architettura prendendo nel dettaglio gli strati caratteristici.

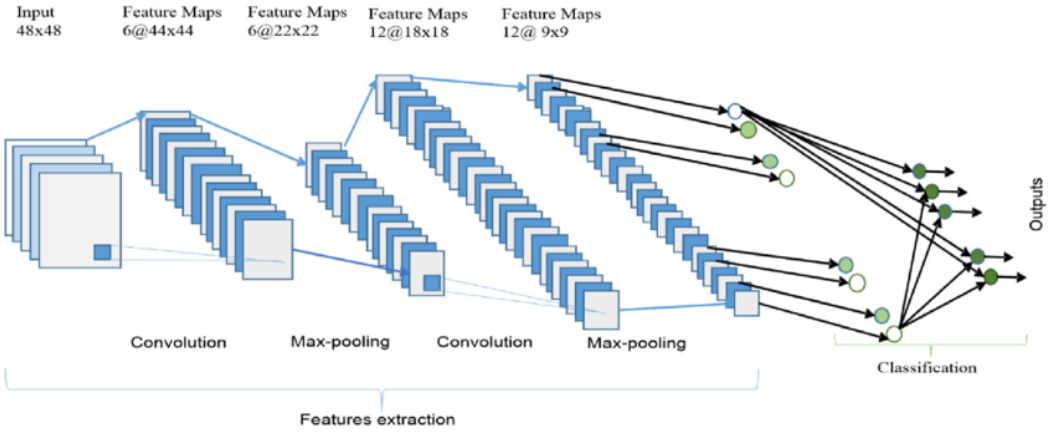

Figura 3.8. Schematizzazione di un Convolutional Neural Network osservando gli strati più significativi.

Innanzitutto, il Convolutional Neural Network è diviso in due sezioni: una parte iniziale dedicata alla feature extraction e una parte finale dedicata alla classificazione vera e propria.

Si inizia ricevendo in input un'immagine e applicando a essa un filtro o convolutional kernel, il quale è di un certo numero di pixel abbastanza piccolo rispetto all'immagine nella sua totalità. Applicare un filtro a tutta l'immagine, analizzandola dunque in sezioni, significa eseguire il prodotto scalare tra la sezione dell'immagine e il filtro, ottenendo un particolare numero come risultato, al quale dovrà essere aggiunto il bias. Applicando questa operazione in ogni sezione si otterranno vari numeri da inserire all'interno di una mappa di features, la quale schematizza l'immagine in una versione più compatta e dalle informazioni significative. Ciò che si è appena descritto è quel che avviene all'interno dello strato di convoluzione e l'operazione nel complesso è infatti detta convoluzione, ciò che dà il nome all'intera rete neurale.

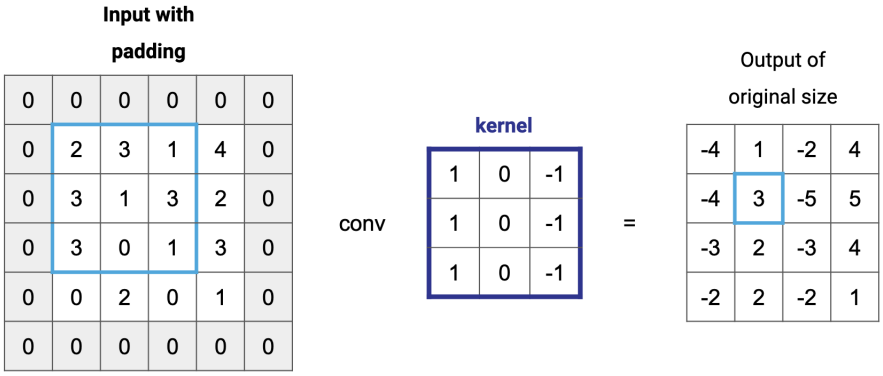

Figura 3.9. L'operazione di convoluzione, ciò su cui si basa la CNN.

Tra lo strato relativo alla convoluzione e del max-pooling, si applica alla mappa di features una funzione di attivazione, solitamente ReLU (3.3).

All'interno dello strato successivo avviene il max-pooling, un'operazione che scorre la mappa di features e per ogni sezione senza sovrapposizione estrae il massimo, creando una mappa modificata.

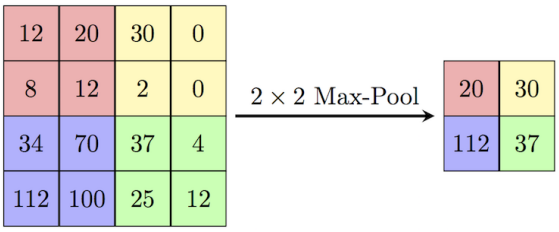

Figura 3.10. L'operazione di max-pooling, applicata su un generico esempio.

Queste due tipologie di strati si ripetono all'interno della CNN quante volte è necessario per ridurre al minimo la mappa di features, completando la parte di feature extraction.

Giunti a questo punto si estrae il contenuto della mappa di features finale e si passano come input a un Multilayer perceptron fully connected, il quale si occuperà della fase di classificazione nella sua totalità. Una volta raggiunte le predizioni è

possibile applicare ai risultati una funzione SoftMax per una migliore comprensione degli stessi.

3.3.2 Recurrent Neural Network

Una rete neurale ricorrente è una rete neurale che presenta come caratteristica e punto di forza la sua architettura diversa da quella generica del feedforward trovata nel MLP e nella CNN. Se nelle altre architetture le informazioni effettuano un solo passaggio e vengono processate immediatamente, nel RNN si effettuano vari passaggi per poter modellare anche l'aspetto temporale di ciò che sta venendo analizzato. Come si può facilmente intuire, una caratteristica di questo tipo può essere fondamentale per il riconoscimento delle azioni negli esseri umani. Sfruttando la componente temporale le sequenze video, o nel nostro caso sequenze di scheletri 3D, la RNN si avvicina di più all'obbiettivo da raggiungere.

Ciò che rende l'RNN capace effettivamente di osservare la componente temporale dell'input è la presenza di un ciclo interno alla rete neurale tra le varie componenti. Questo ciclo permette a un'informazione precedente di influenzare direttamente un'informazione successiva, modellando lo scorrere del tempo e le relazioni tra esse. La Figura 3.11 illustra una generica architettura RNN.

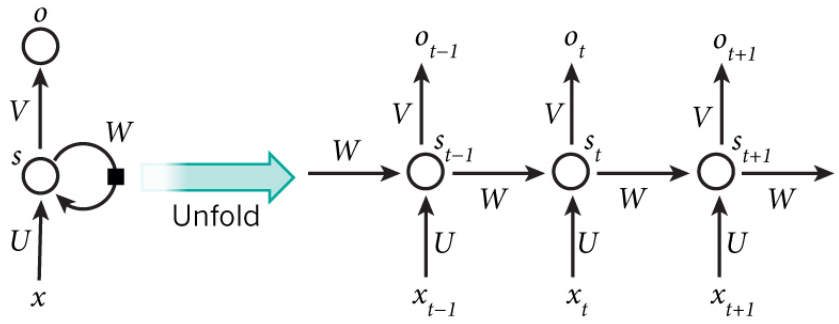

Figura 3.11. Un esempio di RNN, mostrando il ciclo e la sua rappresentazione unrolled.

Ciò che viene mostrato è come in base alla quantità di input nel tempo si può osservare l'RNN come un insieme di copie della rete originale, avendo come particolarità il passaggio di informazioni da una rete all'altra, influenzando al tempo t-1 il tempo t, e così via. Ciò che bisogna notare è come si tratti di copie effettive in cui i pesi e bias rimangono identici rispetto all'originale, così da non dover aumentare il numero di pesi e bias da allenare, rendendo potenzialmente il processo di retropropagazione dell'errore ancora più lento.

Uno dei problemi più gravi che affligge il Recurrent Neural Network, e il motivo per cui allo stato attuale è stato superato da un suo successore che presto vedremo, è l'esplosione/sparizione del gradiente.

Osservando la struttura del RNN ed il passaggio di pesi tra una copia e l'altra, così come la normale retropropagazione, si può osservare come il gradiente (calcolato in precedenza con derivate e pesi) può con l'aumentare dei passaggi diminuire sempre di più fino a zero, oppure aumentare sempre di più esponenzialmente. In entrambi i casi la conseguenza è il rallenamento dell'allenamento della rete neurale

o l'impossibilità a proseguirlo, ritrovandosi a non poter mai raggiungere il minimo locale.

3.3.3 Long-Short Term Memory

Il Long-Short Term Memory (LSTM) è una tipologia di RNN che ha come obbiettivo quello di rimediare ai problemi relativi al gradiente, adottando un'architettura che ha reso obsoleti i semplici RNN. Come il nome della rete neurale suggerisce il LSTM possiede una memoria a breve termine e una memoria a lungo termine. Esse sono le componenti fondamentali, e a loro volta interagiscono tra di loro con tre sezioni distinte chiamate rispettivamente: Forget Gate, Inupt Gate e Output Gate. Proprio come per l'RNN lo LSTM può essere unrolled e visto come numerose unità diverse, in figura Figura 3.12 osserviamo una singola unità nel dettaglio.

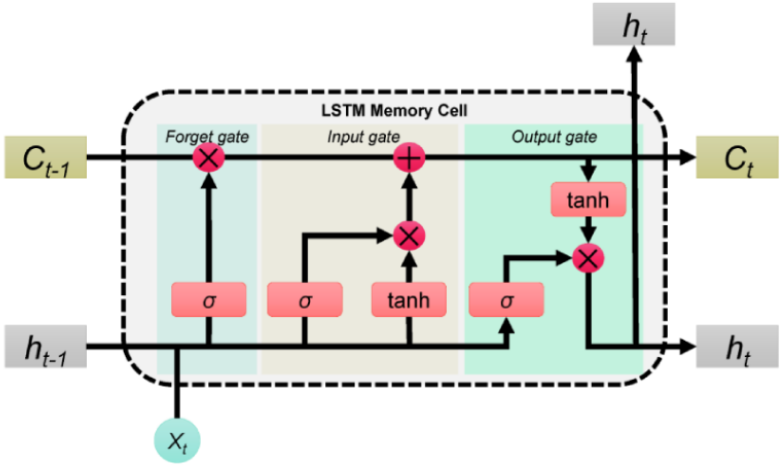

Figura 3.12. In figura è illustrata un'unità del LSTM, evidenziando il forget gate, l'input gate e l'output gate.

Seguendo lo schema, si distingue la linea da cui parte h_{t-1} come la memoria a breve termine, mentre quella con C_{t-1} è detta memoria a lungo termine. Anche se non viene mostrato in figura, la maggiore differenza tra le due memorie è la chiave per evitare il problema del vanishing/exploding gradient: la memoria a breve termine si comporta come il normale RNN, portando con sé i pesi; la memoria a lungo termine invece non ha alcun peso associato, basandosi soltanto sulle interazioni con i vari cancelli.

Proseguendo, i rettangoli rossi presentano rispettivamente il simbolo σ che rappresenta la funzione di attivazione sigmoidea3.2, e la funzione di attivazione tan(h) che mappa i valori tra -1 e 1.

Entrando nel dettaglio si parte dal forget gate, il quale si occupa di gestire quanta percentuale di memoria a lungo termine registrata dovrà essere preservata. Ciò è deciso a partire dall'input X_t oltre che dalla memoria a breve termine h_{t-1}. La somma

tra memoria a breve termine e input con pesi associati viene passata alla funzione sigmoidea, la quale produrrà un risultato da moltiplicare alla memoria a breve termine, determinando quanto quel valore possa essere preservato o dimenticato.

L'input gate decreta quale parte e quanto della memoria a breve termine vada trasformato in memoria a lungo termine. La sezione che sfrutta la funzione sigmoidea decide quanta percentuale della memoria scelta può essere fusa con la memoria a lungo termine, mentre la sezione che usa la funzione tangente decreta una potenziale memoria da aggiungere a quella a lungo termine. Entrambe le sezioni vengono moltiplicate tra loro e sommate alla memoria a lungo termine.

Infine, l'output gate si occupa di aggiornare la memoria a breve termine. La memoria a lungo termine viene passata per la funzione tangente, mentre la memoria a breve termine precedente, con l'input annesso, in quella sigmoidea. Effettuate queste trasformazioni si moltiplicano tra loro producendo la nuova memoria a breve termine h_t.

Avendo analizzato nella sua interezza una sua unità, si ricorda di nuovo che in base alla temporalità dell'input il LSTM può essere unrolled indefinitivamente. Nella figura Figura 3.12 viene mostrata la visione d'insieme finale.

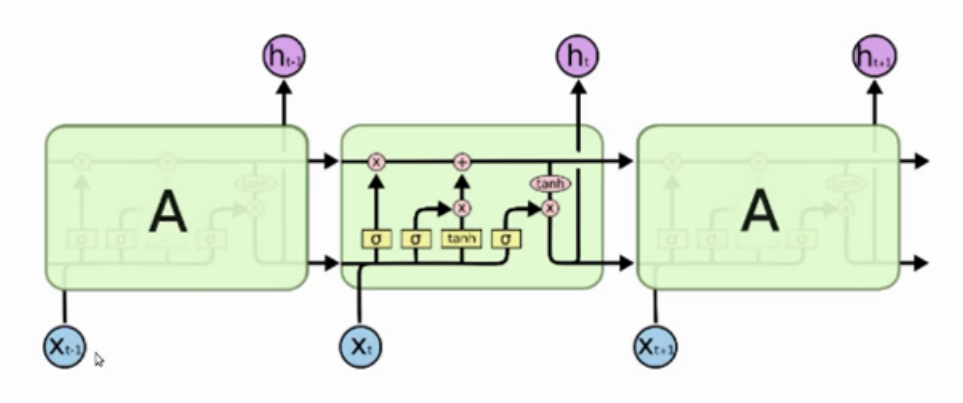

Figura 3.13. La figura illustra una LSTM con più unità, nella sua forma completa.

3.3.4 Graph-convolutional Network

Un Graph-convolutional network (GCN) è una rete neurale che si basa sul concetto di grafo, una struttura matematica composta da nodi ed archi collegati tra loro. Tutte le tipologie di informazioni che possono essere tramutate in grafi possono potenzialmente essere analizzate da una GCN. La struttura di grafo permette di lavorare con una tipologia di dati non euclidea prestandosi ad ambiti altrimenti difficili da studiare, come la correlazione dei post nei social media. Nella figura Figura 3.13 un esempio di struttura generica di GCN.

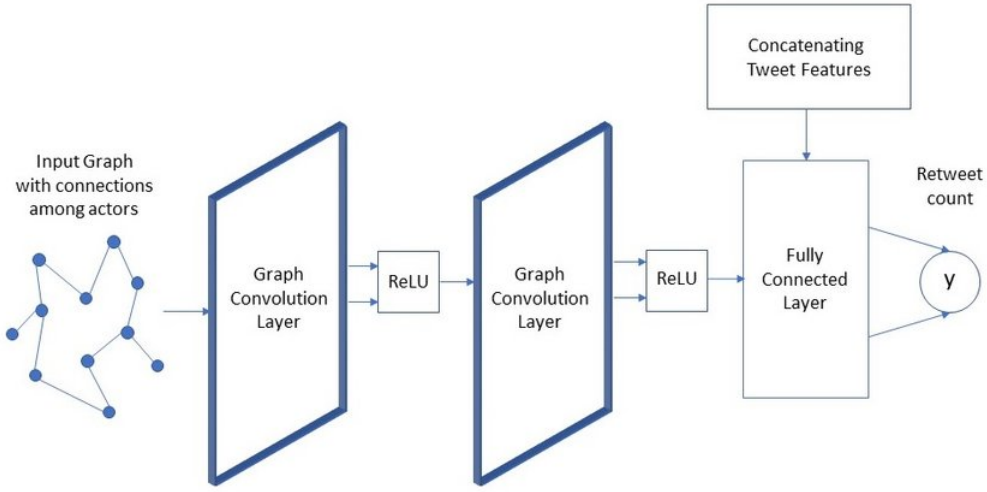

Figura 3.14. Un esempio specifico di Graph-Convolutional Network con una classificazione finale in merito ai tweet per le elezioni.

Una GCN è una sottocategoria di una generica Graph Neural Network (GNN) la quale applica in modo simile alla CNN una convoluzione come principale operazione all'interno della propria struttura.

In una GNN l'idea fondamentale nell'analizzare le features si basa sul concetto di message passing o passaggio di messaggi. La convoluzione previamente menzionata viene sfruttata come tecnica nel message passing, sfruttando la stessa intuizione su cui si basano i modelli di Convolutional Neural Network. In quella casistica la convoluzione aggregava le informazioni di pixel circostanti, nel caso di un GCN il message passing si basa sull'eseguire una convoluzione nel grafo, aggregando informazioni riguardanti nodi vicini.

All'interno dell'architettura mostrata la convoluzione avviene internamente al Graph Convolution Layer, permettendo alla rete neurale di ottenere dai singoli nodi non solo le proprie informazioni personali, ma anche il loro contesto e come si relazionano con l'insieme. Il numero di strati di questo tipo decreta quanto un singolo nodo possa conoscere in profondità il grafo; ogni strato equivale a un passo nel grafo a partire dal nodo preso in considerazione, portando alla fine ad avere ogni nodo influenzato da tutti i nodi del grafo.

Una volta eseguita questa operazione viene applicata una funzione di attivazione, solitamente ReLU, onde evitare che il passaggio di informazioni tra nodi con molti vicini vada fuori controllo, richiedendo quindi una necessaria normalizzazione.

Infine, prima di passare al prossimo Graph-Convolution layer successivo, talvolta è possibile avere a che fare con un pooling layer che in modo simile a come si è visto per la CNN si occuppa di ridurre le dimensioni del grafo per ottenere una rappresentazione più succinta e rappresentativa.

Dopo un numero sufficiente di strati si ottiene un grafo composto da nodi che posseggono informazioni diverse dalle iniziali, avendo in mente il contesto generale. Giunti a questo punto è possibile effettuare varie operazioni in base al tipo di problema. Per il comune problema di classificazione si passa il tutto a un fully connected layer per la predizione finale.

3.4 Conclusioni

All'interno del capitolo si è introdotto il concetto di machine learning, partendo da una visione generica d'insieme e spiegando poi nel dettaglio l'origine e lo sviluppo delle reti neurali artificiali.

Si è partiti dal percettrone e successivamente si è spiegato lo sviluppo fino al Multilayer Perceptron, la struttura basilare delle reti neurali moderne. Se ne è spiegato nel dettaglio il funzionamento menzionando i vari metodi di apprendimento e seguendo nel dettaglio l'intero processo di allenamento della rete neurale con la retropropagazione.

Successivamente si è introdotto il concetto di deep learning e sono state menzionate le architetture più preponderanti, in particolar modo pensando al riconoscimento delle azioni umane con lo scheletro 3D. Tra le varie architetture presentate la GCN attualmente è l'architettura più utilizzata nello stato dell'arte per gestire il task. Lo scheletro umano si presta molto facilmente alla rappresentazione tramite grafo, di conseguenza è chiaro il motivo per cui la GCN ha preso sempre più piede come mostrato negli esempi precedenti.

Capitolo 4

Architettura

Il capitolo è strutturato nel seguente modo: nella sezione 4.1 viene presentata una visione d'insieme dell'architettura del modello; il paragrafo 4.2 si concentra sulla fase di feature extraction e preparazione delle features da far processare al modello; il paragrafo 4.3 mostra la metodologia creata per la rappresentazione dei grafi dello scheletro; nel paragrafo 4.4 si entra nel dettaglio specifico dell'architettura nella sua totalità; infine, nel paragrafo 4.5 si effettuano delle considerazioni sul modello presentato.

4.1 Metodo

Lo studio effettuato si basa sul modello [39] di Lee et al. detto HD-GCN o Hierarchically Decomposed Graph Convolutional Network.

Il metodo utilizzato prende in input una sequenza di scheletri 3D utilizzando varie tecniche di feature extraction per ottenere un risultato più consistente, come molti altri modelli in letteratura. Il risultato di queste operazioni viene passato alla rete neurale, la quale è composta da nove blocchi GCN (Graph-Convolutional Network) consequenziali, in seguito a ciò è presente uno strato completamente connesso, attraverso il quale si ha una funzione di pooling e infine una funzione di softmax per la classificazione finale dell'azione eseguita.

I blocchi GCN menzionati sono a loro volta composti da un modulo spaziale ed un modulo temporale consecutivi e come verrà mostrato con più precisione in seguito il modulo spaziale è diviso in una HD-Graph convolution e un modulo dell'attenzione.

Nella Figura 4.1 viene mostrata l'intera architettura della rete neurale utilizzata per il riconoscimento delle azioni tramite scheletro 3D.

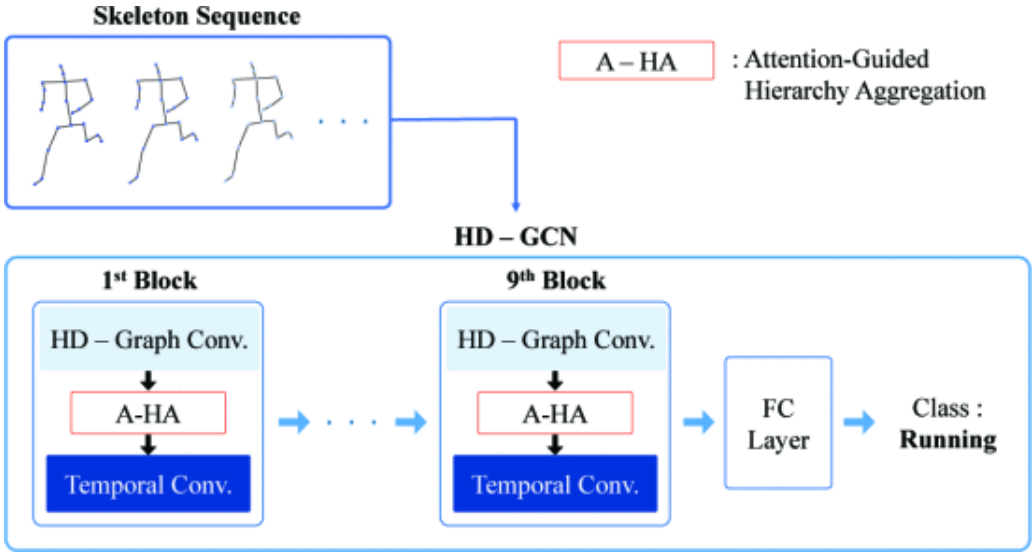

Figura 4.1. L'architettura dell'HD-GCN, ottenuta dal paper di riferimento [39].

4.2 Feature extraction

A partire dal dataset di riferimento, il quale verrà analizzato successivamente, è necessario estrarre le informazioni rilevanti per analizzare la singola sequenza scheletrica, corrispondente a una sola azione performata.

Il metodo utilizzato prevede per ogni singolo file skeleton di estrarre le informazioni relative alle articolazioni, per ogni diversa persona presente nella scena. Il risultato finale per questo primo passo del processo è un dizionario che fa riferimento a una particolare sequenza scheletrica, riportando le informazioni estratte e il numero di frame presenti nella sequenza. Le informazioni prese sono a loro volta un dizionario contenente la posizione 3D delle articolazioni, la posizione 2D dei colori, una lista di frames e un ulteriore valore motion per le sequenze con più di un partecipante.

Successivamente, si effettuano varie operazioni per eliminare il rumore presente nelle informazioni ottenute. Innanzitutto si eliminano le sequenze che non presentano abbastanza frame validi come scelto nel passo precedente e oltre a ciò si effettua un'analisi sulla rumorosità delle informazioni, eliminando ciò che supera una certa soglia di rumore. Si osserva il valore motion e si filtrano le sequenze che superano un certo intervallo. Prima di effettuare le modifiche finali si eliminano tutti i frame senza alcun individuo, procedendo poi a creare per ogni frame di una sequenza scheletrica un vettore contenente le posizioni di tutte le articolazioni di ogni rispettivo partecipante.

Il passo finale prima di effettuare le divisioni dei dati per allenamento, validazione e testing prende in input ciò che si è ottenuto fino a ora, traslando le posizioni delle articolazioni in base a una specifica tra esse, reimpostando l'origine delle coordinate. Fatto ciò si conclude allineando dal punto di vista dei frames presenti tutte le sequenze, riempiendo di zeri le sequenze più corte, restituendo le informazioni nel loro stato finale.

4.3 Grafo gerarchicamente decomposto

Come già detto in precedenza lo scheletro umano con le articolazioni si presta bene a essere rappresentato tramite un grafo, infatti l'architettura si basa sul Graph-Convolutional Network (GCN). Solitamente in metodi di questo tipo le connessioni tra articolazioni vengono create manualmente, mentre nel metodo proposto per l'HD-GCN si ha un grafo gerarchicamente decomposto.

Come primo passo per la decomposizione è necessario scegliere un centro della massa (Center of Mass) da cui dipenderà il resto del grafo. Prendendo il CoM si andrà a creare un albero radicato, avendo come radice il CoM e per ogni strato successivo tutti gli archi alla n_{th} distanza dal CoM, creando una serie di insiemi gerarchici finitamente connessi, come mostrato nella prima parte della Figura 4.2.

Una volta fatto ciò è necessario modellare la relazione tra articolazioni distanti, soprattutto nello stesso spazio semantico. In questo modo creiamo degli archi completamernte connessi tra tutti i nodi e tra gli insiemi gerarchici vicini, aumentando notevolmente il proprio campo ricettivo, come mostrato nella seconda parte della Figura 4.2.

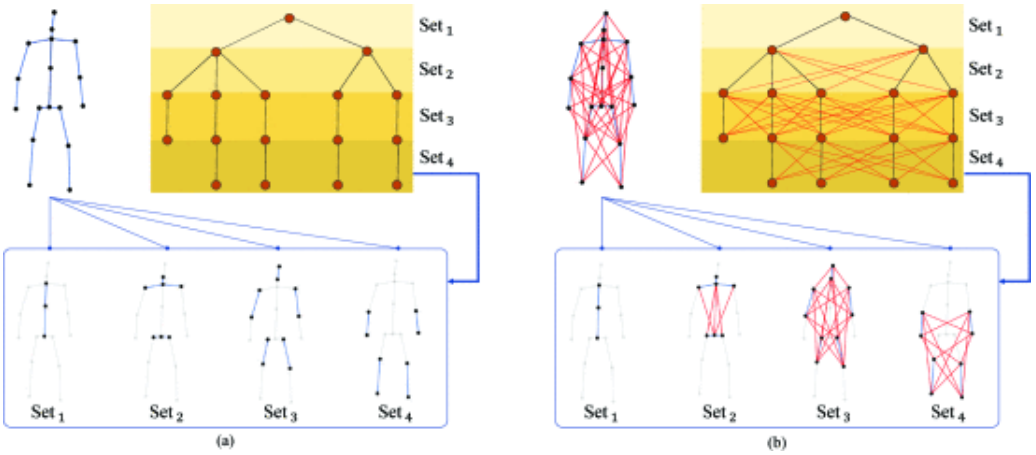

Figura 4.2. A sinistra la rappresentazione dello scheletro con archi fisicamente connessi, mentre a destra in blu si hanno gli archi fisicamente connessi e in rosso gli archi completamente connessi, ottenuti dal paper di riferimento [39].

4.4 Architettura

Come precedentemente menzionato, l'architettura di HD-GCN si basa su tre blocchi principali ripetuti per nove volte, fino a un fully connected layer finale. Si analizzeranno ora i tre blocchi separatamente, partendo dalla parte spaziale, con la convoluzione su grafo e l'aggregazione gerarchica guidata dall'attenzione, e finendo con la parte temporale.

La prima parte del blocco spaziale è la convoluzione su grafo, la quale viene effettuata su ogni singolo insieme di archi gerarchico e poi concatenata in un unico risultato. Per un singolo insieme si hanno quattro operazioni parallele: tre convoluzioni su grafo e un'operazione detta EdgeConv. La formula per una singola convoluzione è la seguente:

$$F_{HD}^{(k)} = \|_{s \in S} \left\{ A_{HD_{;s}}^{\leftrightarrow(k)} \Phi(F_{in}) \Theta_s^{(k)} \right\}$$

A ciò che si è ottenuto dai quattro rami si applica una trasformazione lineare e alla fine una concatenazione degli output, producendo come risultato una feature map. Analizzando più nello specifico il ramo che esegue l'operazione EdgeConv si può vedere come inizialmente viene effettuata un'operazione di average pooling sulla dimensione temporale, riducendo la dimensionalità. Entrando in EdgeConv bisogna considerare come la struttura di grafo precedentemente costruita qui venga ignorata, lasciando che sia il modello stesso a creare le connessioni tra nodi. Per scegliere come creare questi archi si osservano le somiglianze e la distanza minore tra k-nodi vicini. Fatto ciò si aggrega con una convoluzione, riportando il risultato finale del ramo. Questo intero processo si può osservare nella parte destra della Figura 4.3.

La formula associata che riassume l'intero processo per un singolo insieme è la seguente, dove z_v rappresenta EdgeConv spaziale:

$$F_{HD} \leftarrow \sum_{k=1}^{N_L} \left[F_{HD}^{(k)} \parallel z_v^{(k)} \left(\frac{1}{T} \sum_{t=1}^{T} \Phi(F_{in}) \right) \right]$$

Una volta effettuata la HD-Graph Convolution sugli insiemi ed ottenuto un singolo risultato concatenato si passa alla seconda parte del blocco spaziale con un meccanismo di attenzione. L'obbiettivo dietro questa scelta è la possibilità di dare una maggiore attenzione dinamicamente a una parte di archi presenti nel grafo, in base alla specifica azione. Nella parte destra dell'immagine Figura 4.3 sono presenti nel dettaglio le operazioni che ora verranno analizzate.

Innanzitutto si effettua un'operazione di Max Pooling, successivamente si applica il modulo detto RSAP (Representative Spatial Average Pooling). Al suo interno esso si occupa di osservare i vari nodi in un certo insieme gerarchico, scegliendo per ognuno il suo massimo nella sequenza temporale. Una volta fatto ciò si effettua una media su tutti i vettori risultanti appartenenti ai nodi, ottenendo così un singolo vettore rappresentante di tutto l'insieme gerarchico di archi. La seguente formula riassume RSAP, dove N_k è il numero di vertici nell'insieme H_k:

$$\Psi(F_{HD}^{(k)}) = \frac{1}{N_k + N_{k+1}} \sum_{v \in H_k \cup H_{k+1}} \max_t \left(F_{HD}^{(k)}(v) \right)$$

Come ultimo passo prima della somma e del risultato finale da passare al blocco temporale, viene effettuata una EdgeConv come definita precedentemente, applicando una funzione sigmoidea. Qui sotto è presente la formula che produce una mappa dell'attenzione:

$$M = \sigma \left(z_L \left(\|_{k \in L} \left\{ \Psi(F(k)^{HD}) \right\} \right) \right)$$

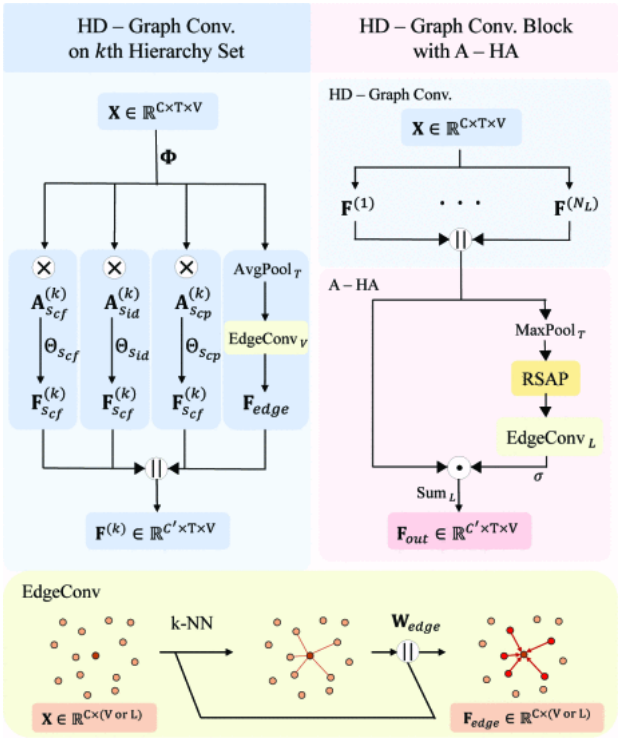

Figura 4.3. A sinistra l'operazione di convoluzione sul grafo per un singolo insieme, in basso l'operazione di EdgeConv e a destra l'operazione di convoluzione su tutti gli insiemi con il meccanismo di attenzione, a formare l'intero blocco spaziale. Preso dal paper di riferimento [39].

Il blocco temporale invece consiste in quattro diversi rami paralleli di operazioni. Due di questi sono delle convoluzioni temporali dilatate con una dimensione del kernel di cinque, con una dilatazione rispettivamente di uno e due. Ciò che rimane è una convoluzione punto a punto e un max pooling con una dimensione del kernel pari a 3.

Ripetendo il modulo spaziale e quello temporale per 9 volte si ottiene l'architettura completa, ottenendo con uno strato completamente connesso la classificazione finale.

4.5 Conclusioni

Nel capitolo si è trattato dell'architettura del modello nel dettaglio, introducendo inizialmente una visione d'insieme. Si è partiti con un'analisi dell'estrazione e trasformazione di features a partire dai dati grezzi fino al suo inserimento nel modello vero e proprio.

Successivamente si è suddivisa l'architettura in blocco spaziale e blocco temporale, spiegando sequenzialmente il passaggio di dati e le loro trasformazioni. Compresa la convoluzione su grafo, il meccanismo di attenzione e la convoluzione temporale si è concluso con lo strato completamente connesso e il risultato ultimo.

Capitolo 5

Esperimenti e Risultati

Il capitolo è strutturato nel seguente modo: nella sezione 5.1 viene analizzato il dataset scelto, spiegando nel dettaglio la sua composizione; il paragrafo 5.2 presenta la macchina utilizzata per i test e gli iperparametri scelti; nel paragrafo 5.3 si effettua un confronto con il risultato ottenuto e i vari metodi allo stato dell'arte; infine, nel paragrafo 5.4 vengono tratte delle conclusioni.

5.1 Dataset

Il dataset utilizzato per le varie fasi di training, validation e testing del modello è l'NTU RGB+D 60 [40]. Nella letteratura questo è uno dei dataset più utilizzati in assoluto ed è anche il primo ad avere una grande quantità di dati (più di 56.000) riguardanti lo scheletro 3D. Come il nome del dataset stesso può suggerire le informazioni registrate sono video RGB, sequenze di profondità, informazioni 3D sullo scheletro e sequenze infrarossi. All'interno del dataset sono presenti 60 classi di azioni differenti, giornaliere o relative alla salute, eseguite da 40 individui diversi tra loro per quanto riguarda altezza ed età, spaziando dai 10 ai 35 anni. Oltre a ciò, tre diversi punti di vista delle telecamere utilizzati offrono una variazione di prospettiva della stessa azione ed una complessità aggiunta nell'allenamento. I punti di vista principali presentano tre diversi angoli orizzontali: -45°, 0° e 45°. Per quanto le angolazioni delle viste siano tre principalmente, il numero finale è più vicino a 80, contando anche cambi di altezza e distanza da chi esegue l'azione, per diversificare ulteriormente.

Per quanto riguarda le informazioni che sono effettivamente rilevanti, per questo studio sono le sequenze scheletriche per ogni video. Un singolo file scheletrico presenta per ogni frame del video la posizione di 25 articolazioni del corpo. In un singolo video sono presenti al massimo due persone contemporaneamente.

Alla luce di quanto detto, quando ci si riferisce al dataset si distingue tra due diversi approcci: Cross Subject e Cross View. La modalità Cross View divide per il proprio allenamento e testing in base a una delle 3 principali posizioni delle telecamere. La modalità Cross Suject invece divide il dataset in base agli individui che eseguono le azioni, ai quali è assegnato un id univoco attraverso ogni azione e punto di vista differente; questa è la modalià su cui ci si è concentrati.

5.2 Dettagli di implementazione

Una fase sperimentale è stata effettuata per migliorare il modello scelto. Tutti gli esperimenti sono stati eseguiti su una CPU 12-Core Intel i7-12700K con 3.60 GHz e 32GB di RAM; la GPU utilizzata è una Nvidia GeForce RTX 3060. Le principali librerie e programmi usati in Python sono: PyTorch, PyYAML, matplotlib, einops, sklearn, tqdm, tesnorboardX, h5py e torchlight.

Per quanto riguarda gli iperparametri della rete neurale, si è utilizzato un numero di epoche pari a 90 con 5 epoche di warm up con un base learning rate di 0.1. Il learning rate sfruttato presenta un massimo di 1.0 ed un minimo di 0.0001. Dal punto di vista dell'optimizer si è sfruttato lo Stochastic Gradient Descent con un Nesterov momentum di 0.9 e weight decay di 0.0004. La batch size utilizzata è pari a 28.

Durante soltanto la fase di training è stata applicata ai dati una rotazione casuale per ottenere una generalizzazione migliore e per applicarsi in modo migliore a casistiche reali.

Il metodo utilizzato come altri in letteratura sfrutta una metodologia detta ensamble, la quale prende risultati da diversi allenamenti con caratteristiche diverse unendo i risultati e ottenendo un'accuratezza maggiore.

Rispetto al modello originale, il quale sfruttava il 50% dei dati per il training e il 50% per test e validazione, la modifica effettuata in questo lavoro è più significativa ed è anche più coerente con la filosofia dietro il machine learning, eseguendo un test finale su dati mai visti prima. Lo split applicato al datset assegna l'80% dei dati alla fase di training, 10% dei dati alla fase di validation e 10% alla fase di testing su dati mai toccati fino a quel momento. Il motivo per cui è stato effettuato questo split è la grande quantità di dati a disposizione del dataset, infatti anche in questo modo non si è sofferto di overfitting. Dal punto di vista della metodologia Cross Subject si sono divisi in queste tre parti gli id dei 40 partecipanti del dataset. Gli id 6, 11, 24 e 37 sono stati assegnati alla fase di validation; gli id 3, 12, 26 e 30 sono stati usati per il test finale, mentre i restanti sono stati usati per il training.

I risultati ottenuti con questi parametri sono pari a un'accuratezza top 1 del 93.87%, superando il paper originale e una buona parte dei metodi usati nello stato dell'arte.

5.3 Confronto con lo stato dell'arte

La tabella 5.1 mostra un confronto tra il modello da cui si è partiti per questo lavoro, l'HD-GCN, gli altri modelli presenti nello stato dell'arte e il modello di questo lavoro con le modifiche apportate.

Tabella 5.1. Confronto tra l'HD-GCN, *la nostra modifica** e lo stato dell'arte, per il dataset NTURGB+D60 nella modalità Cross Subject

NTU-RGB+D 60 (cs)	Accuratezza (%)
Wang et al.[41] (ViT-L)	94.3
Duan et al.[42]	94.1
Wang et al.[41] (ViT-B)	94
*HD-GCN modificato**	93.8
Zhang et al. [43]	93.7
Do et al.[44]	93.5
Liu et al.[45]	93.5
Xu et al.[46]	93.5
Lee et al.[39] (HD-GCN)	93.4
Duan et al.[47]	93.2

Come si può osservare dalla tabella, il nuovo metodo non supera in assoluto lo stato dell'arte, ma è un netto miglioramento sull'originale, sorpassando nel processo numerosi metodi in quanto ad accuratezza.

5.4 Conclusioni

Nel capitolo si è spiegato nel dettaglio la struttura del dataset scelto con tutte le sue particolarità e caratteristiche, successivamente si è approfondito dal punto di vista tecnico i parametri utilizzati e le scelte per la suddivisione dei dati, il punto fondamentale per il risultato ottenuto. Per concludere: si è fatto un confronto con lo stato dell'arte, raggiungendo l'obbiettivo prefissato e sorpassando il metodo originale in quanto ad accuratezza.

Capitolo 6

Conclusioni

In questo lavoro di tesi si è analizzato lo stato dell'arte riguardante il riconoscimento delle azioni e si è scelta tra le varie modalità di dati quella del 3D Skeleton per via delle sue peculiari proprietà e vantaggi. Comprendendo come l'architettura del Graph-Convolutional Network fosse la più adatta a descrivere lo scheletro umano, si sono analizzate nella letteratura le tecniche di machine learning più vicine allo stato dell'arte. Scegliendo come punto di partenza l'architettura detta HD-GCN si è deciso di provare a migliorare ciò che si aveva a disposizione, per ottenere una maggiore accuratezza. Analizzandone la struttura e modificandone in parte gli iperparametri si è osservato come senza modificare l'architettura in sé per sé, ma agendo sulle modalità di implementazione, si è riusciti a raggiungere un risultato migliore, soddifacendo l'obbiettivo iniziale.

Per quanto riguarda potenziali sviluppi dell'ambito e di questo lavoro in particolare, potenzialmente si può pensare di eseguire ulteriori trasformazioni con le features per ottenere dei tempi di allenamento e analisi inferiori agli attuali, estremamenti lunghi. Oltre a ciò, con l'avanzare del tempo è possibile che nell'ambito dei Graph Convolutional Network possano idearsi dei moduli temporali migliori a quanto proposto, in quanto quello utilizzato è semplicemente riproposto a partire da modelli passati. Infine, sull'intera branca dello Human Action Recognition negli anni a venire, al posto dei GCN i transformers potrebbero potenzialmente diventare predominanti.

Bibliografia

[1] Y. Ji, Y. Yang, F. Shen, H. T. Shen and X. Li, *A Survey of Human Action Analysis in HRI Applications*, in IEEE Transactions on Circuits and Systems for Video Technology, vol. 30, no. 7, pp. 2114-2128, July 2020

[2] J. Huang, S. Lin, N. Wang, G. Dai, Y. Xie and J. Zhou, *TSE-CNN: A Two-Stage End-to-End CNN for Human Activity Recognition*, in IEEE Journal of Biomedical and Health Informatics, vol. 24, no. 1, pp. 292-299, Jan. 2020

[3] V. D. Huszár, V. K. Adhikarla, I. Négyesi and C. Krasznay, *Toward Fast and Accurate Violence Detection for Automated Video Surveillance Applications*, in IEEE Access, vol. 11, pp. 18772-18793, 2023

[4] F. Wu et al., *A Survey on Video Action Recognition in Sports: Datasets, Methods and Applications*, in IEEE Transactions on Multimedia, vol. 25, pp. 7943-7966, 2023

[5] Z. Sun, Q. Ke, H. Rahmani, M. Bennamoun, G. Wang and J. Liu, *Human Action Recognition From Various Data Modalities: A Review*, in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 45, no. 3, pp. 3200-3225, 1 March 2023

[6] C. Wang and J. Yan, *A Comprehensive Survey of RGB-Based and Skeleton-Based Human Action Recognition*, in IEEE Access, vol. 11, pp. 53880-53898, 2023

[7] X. Weiyao, W. Muqing, Z. Min and X. Ting, *Fusion of Skeleton and RGB Features for RGB-D Human Action Recognition*, in IEEE Sensors Journal, vol. 21, no. 17, pp. 19157-19164, 1 Sept.1, 2021

[8] Khurram Soomro, Amir Roshan Zamir and Mubarak Shah, *UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild*, in arXiv:1212.0402, 3 Dec. 2012

[9] T. Su, H. Wang, Q. Qi, L. Wang and B. He, *Transductive Learning With Prior Knowledge for Generalized Zero-Shot Action Recognition*, in IEEE Transactions on Circuits and Systems for Video Technology, vol. 34, no. 1, pp. 260-273, Jan. 2024

[10] L. Wang, D. Q. Huynh and P. Koniusz, *A Comparative Review of Recent Kinect-Based Action Recognition Algorithms*, in IEEE Transactions on Image Processing, vol. 29, pp. 15-28, 2020

[11] R. Gu, G. Wang, Z. Jiang and J. -N. Hwang, *Multi-Person Hierarchical 3D Pose Estimation in Natural Videos*, in IEEE Transactions on Circuits and Systems for Video Technology, vol. 30, no. 11, pp. 4245-4257, Nov. 2020

[12] C. Bian, W. Feng, L. Wan and S. Wang, *Structural Knowledge Distillation for Efficient Skeleton-Based Action Recognition*, in IEEE Transactions on Image Processing, vol. 30, pp. 2963-2976, 2021

[13] L. Xu, C. Lan, W. Zeng and C. Lu, *Skeleton-Based Mutually Assisted Interacted Object Localization and Human Action Recognition*, in IEEE Transactions on Multimedia, vol. 25, pp. 4415-4425, 2023

[14] S. Yang, J. Liu, S. Lu, E. M. Hwa, Y. Hu and A. C. Kot, *Self-Supervised 3D Action Representation Learning With Skeleton Cloud Colorization*, in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 46, no. 1, pp. 509-524, Jan. 2024

[15] J. Liu, A. Shahroudy, M. Perez, G. Wang, L. -Y. Duan and A. C. Kot, *NTU RGB+D 120: A Large-Scale Benchmark for 3D Human Activity Understanding*, in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 42, no. 10, pp. 2684-2701, 1 Oct. 2020

[16] R. Vemulapalli, F. Arrate and R. Chellappa, *Human Action Recognition by Representing 3D Skeletons as Points in a Lie Group*, 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 2014, pp. 588-595

[17] J. Wang, Z. Liu, Y. Wu and J. Yuan, *Learning Actionlet Ensemble for 3D Human Action Recognition*, in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 36, no. 5, pp. 914-927, May 2014

[18] Y. Du, Y. Fu and L. Wang, *Skeleton based action recognition with convolutional neural network*, 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR), Kuala Lumpur, Malaysia, 2015, pp. 579-583

[19] J. Tu, M. Liu and H. Liu, *Skeleton-Based Human Action Recognition Using Spatial Temporal 3D Convolutional Neural Networks*, 2018 IEEE International Conference on Multimedia and Expo (ICME), San Diego, CA, USA, 2018, pp. 1-6

[20] H. Duan, Y. Zhao, K. Chen, D. Lin and B. Dai, *Revisiting Skeleton-based Action Recognition*, 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 2022, pp. 2959-2968

[21] Yong Du, W. Wang and L. Wang, *Hierarchical recurrent neural network for skeleton based action recognition*, 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 2015, pp. 1110-1118

[22] B. Mahasseni and S. Todorovic, *Regularizing Long Short Term Memory with 3D Human-Skeleton Sequences for Action Recognition*, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 3054-3062

[23] R. Friji, H. Drira, F. Chaieb, H. Kchok and S. Kurtek, *Geometric Deep Neural Network using Rigid and Non-Rigid Transformations for Human Action Recognition*, 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 2021, pp. 12591-12600

[24] S. Yan, Y. Xiong and D. Lin.. *Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition*, in Proc. 32nd AAAI Conf. Artif. Intell., 2018, pp. 7444-7452

[25] T. Ahmad, L. Jin, X. Zhang, S. Lai, G. Tang and L. Lin, *Graph Convolutional Neural Network for Human Action Recognition: A Comprehensive Survey*, in IEEE Transactions on Artificial Intelligence, vol. 2, no. 2, pp. 128-145, April 2021

[26] Wentian Xin, Ruyi Liu, Yi Liu, Yu Chen, Wenxin Yu, Qiguang Miao, *Transformer for Skeleton-based action recognition: A review of recent advances*, Neurocomputing, Volume 537, 2023,Pages 164-186, ISSN 0925-2312

[27] W. Myung, N. Su, J. -H. Xue and G. Wang, *DeGCN: Deformable Graph Convolutional Networks for Skeleton-Based Action Recognition*, in IEEE Transactions on Image Processing, vol. 33, pp. 2477-2490, 2024

[28] P. Zhang, C. Lan, J. Xing, W. Zeng, J. Xue and N. Zheng, *View Adaptive Neural Networks for High Performance Skeleton-Based Human Action Recognition*, in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 41, no. 8, pp. 1963-1978, 1 Aug. 2019

[29] https://github.com/CMU-Perceptual-Computing-Lab/openpose

[30] K. Sun, B. Xiao, D. Liu and J. Wang, *Deep High-Resolution Representation Learning for Human Pose Estimation*, 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 2019, pp. 5686-5696

[31] Microsoft Corp. Redmond WA. Kinect for Xbox 360. 1297, 1298

[32] J. Shotton et al., *Real-time human pose recognition in parts from single depth images," CVPR 2011, Colorado Springs*, CO, USA, 2011, pp. 1297-1304

[33] A. Ramagiri et al., *Image Classification for Optimized Prediction of Leukemia Cancer Cells using Machine Learning and Deep Learning Techniques*, 2023 International Conference on Innovative Data Communication Technologies and Application (ICIDCA), Uttarakhand, India, 2023, pp. 193-197

[34] A. PP, D. Sharma and C. K R, *KIKO: A Tools Framework for Industrial Domain-specific Natural Language Processing Tasks*, 2023 4th International Conference on Communication, Computing and Industry 6.0 (C216), Bangalore, India, 2023, pp. 01-06

[35] S. Jayaraman, M. Ramachandran, R. Patan, M. Daneshmand and A. H. Gandomi, *Fuzzy Deep Neural Learning Based on Goodman and Kruskal's Gamma*

for Search Engine Optimization, in IEEE Transactions on Big Data, vol. 8, no. 1, pp. 268-277, 1 Feb. 2022

[36] Sur et al. (2023). *Supervised versus unsupervised approaches to classification of accelerometry data.* Ecology and Evolution. 13. 10.1002/ece3.10035.

[37] Rukshan Pramoditha, *The concept of Artificial Neurons (perceptrons) in neural networks*, in Towards Data Science, Dec. 26 2021

[38] C. M. Bishop, Neural Networks for Pattern Recognition. New York, NY, USA:Oxford University Press, Inc., 1995

[39] J. Lee, M. Lee, D. Lee and S. Lee, *Hierarchically Decomposed Graph Convolutional Networks for Skeleton-Based Action Recognition*, 2023 IEEE/CVF International Conference on Computer Vision (ICCV), Paris, France, 2023, pp. 10410-10419

[40] A. Shahroudy, J. Liu, T. -T. Ng and G. Wang, *NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis*, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 1010-1019

[41] Y. Wang, Y. Wu, S. Tang, W. He, X. Guo, F. Zhu, L. Bai, R. Zhao, J. Wu, T. He, W. Ouyang, *Hulk: A Universal Knowledge Translator for Human-Centric Tasks*, arXiv:2312.01697 [cs.CV], 2023

[42] H. Duan, Y. Zhao, K. Chen, D. Lin, B. Dai, *Revisiting Skeleton-based Action Recognition*, arXiv:2104.13586 [cs.CV], 2022

[43] J. Zhang, L. Lin, J. Liu, *Shap-Mix: Shapley Value Guided Mixing for Long-Tailed Skeleton Based Action Recognition*, arXiv:2407.12312 [cs.CV], 2024

[44] J. Do, M. Kim, *SkateFormer: Skeletal-Temporal Transformer for Human Action Recognition*, arXiv:2403.09508 [cs.CV], 2024

[45] J. Liu, C. Chen, M. Liu, *Multi-Modality Co-Learning for Efficient Skeleton-based Action Recognition*, arXiv:2407.15706 [cs.CV], 2024

[46] H. Xu, Y. Gao, Z. Hui, J. Li, X. Gao, *Language Knowledge-Assisted Representation Learning for Skeleton-Based Action Recognition*, arXiv:2305.12398 [cs.CV], 2023

[47] H. Duan, J. Wang, K. Chen, D. Lin, *DG-STGCN: Dynamic Spatial-Temporal Modeling for Skeleton-based Action Recognition*, arXiv:2210.05895 [cs.CV], 2022